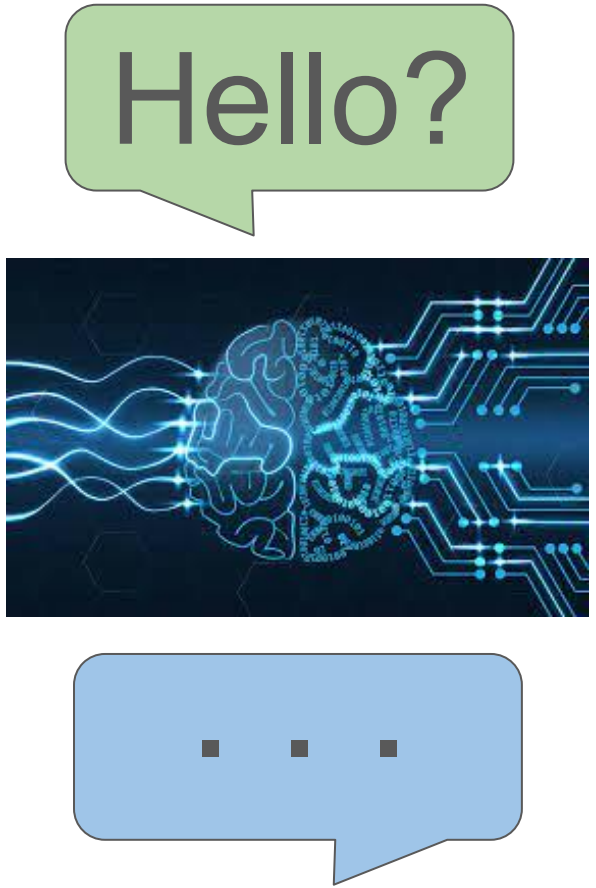
An Introduction to Large Language Models (LLMs)

Brent McPherson, PhD

2025-05-30

Topics

- What are Large Language Models?
- The Impact of LLMs
- Using an LLM Programmatically
  - Tips for getting started
  - RAGs, Agents, and MCP
- What models should we use?

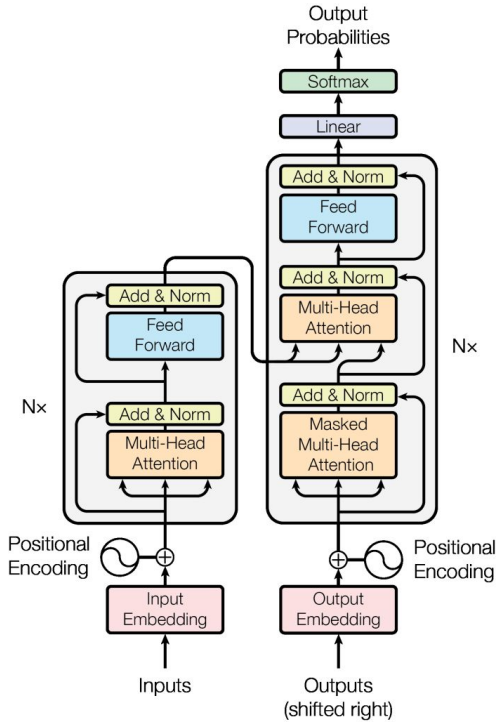


https://www.forbes.com/sites/garydrenik/2023/01/11/large-language-models-will-define-artificial-intelligence/

What are Large Language Models?

What are Large Language Models (LLMs)?

- Massive (billions of parameters) deep learning models.
  - Mostly transformers (encoder / decoder)
- They are trained on a massive corpus (trillions of words) to probabilistically generate responses.
- Conventionally, they provide a plain language interface to interact with them.
  - No “coding” required



https://www.wisecube.ai/blog/a-comprehensive-overview-of-large-language-models/

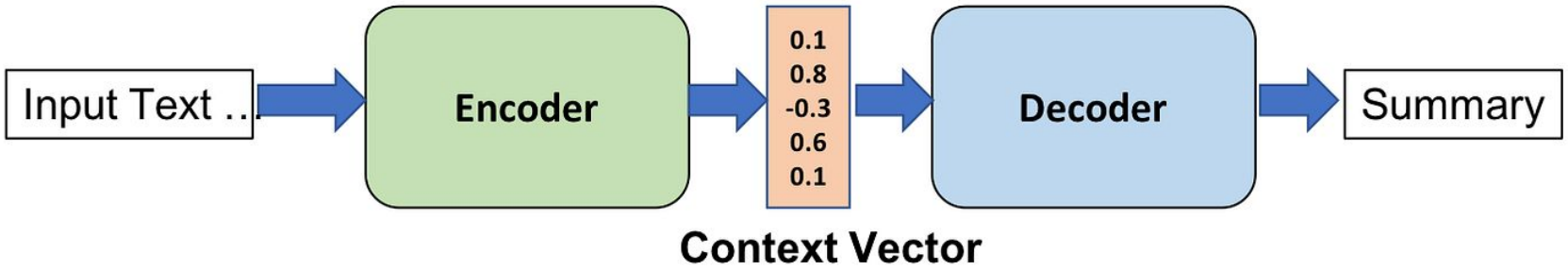
The Parts of an LLM

- Tokenization
  - splitting the input / output texts into smaller units that can be processed by the LLMs.
- Embedding
  - a high dimensional encoding of tokens that represents their semantic meaning.
- Attention
  - LLMs selectively weight the importance of words within the context of their embeddings.
- Pre-training
  - unsupervised training stage of an LLM on large amounts of text to establish word embeddings.
- Transfer Learning
  - utilize the pre-trained weights to provide context to more specialized or fine-tuned data.

What are Large Language Models (LLMs)?

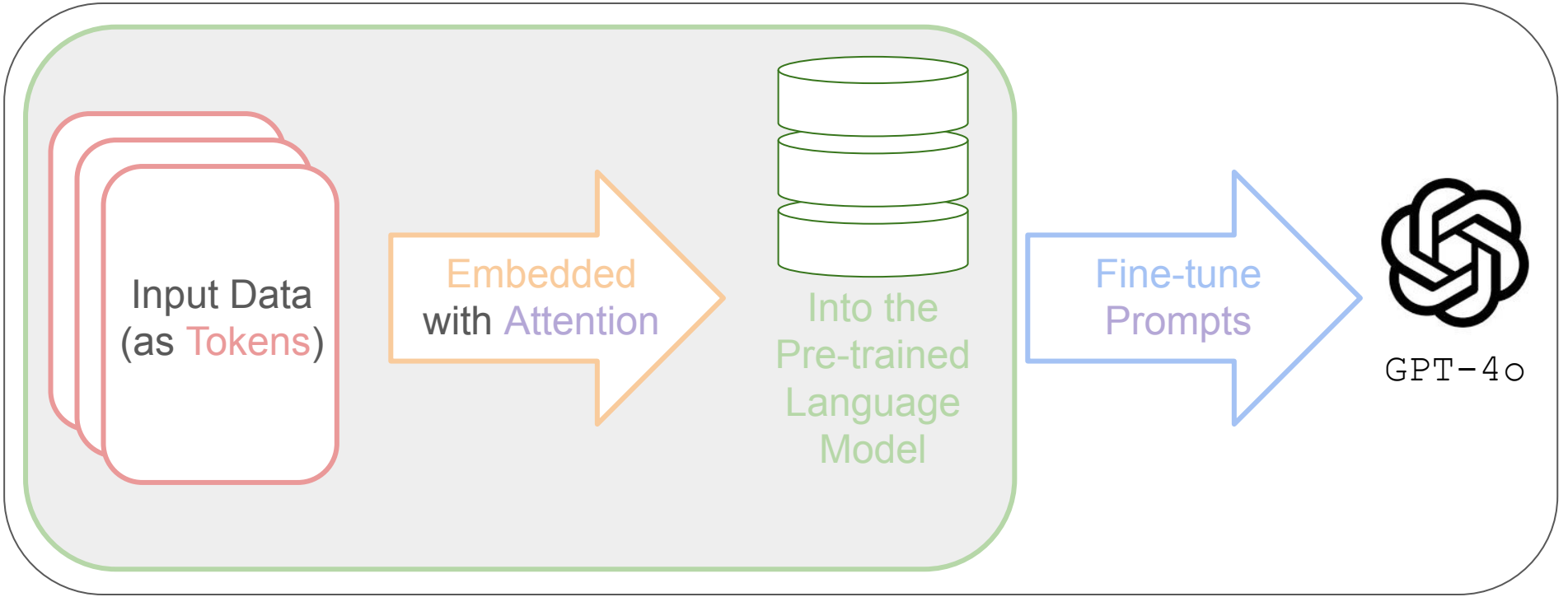
- A transformer model embeds (encodes) the data in a high dimensional space.
- To generate responses, the prompt is parsed into that embedding space to identify relevant information that is based on weights from the training data.
- The kinds of responses are fine-tuned with more targeted kinds of training inputs.
- The response is generated by providing the next most likely segment (token) of text.

Tokenization - Embedding - Attention - Pre-Training - Transfer Learning



https://medium.com/@anishnama20/exploring-the-power-of-encoder-decoder-models-pros-cons-and-applications-8bfbe2e66e76

Building an LLM



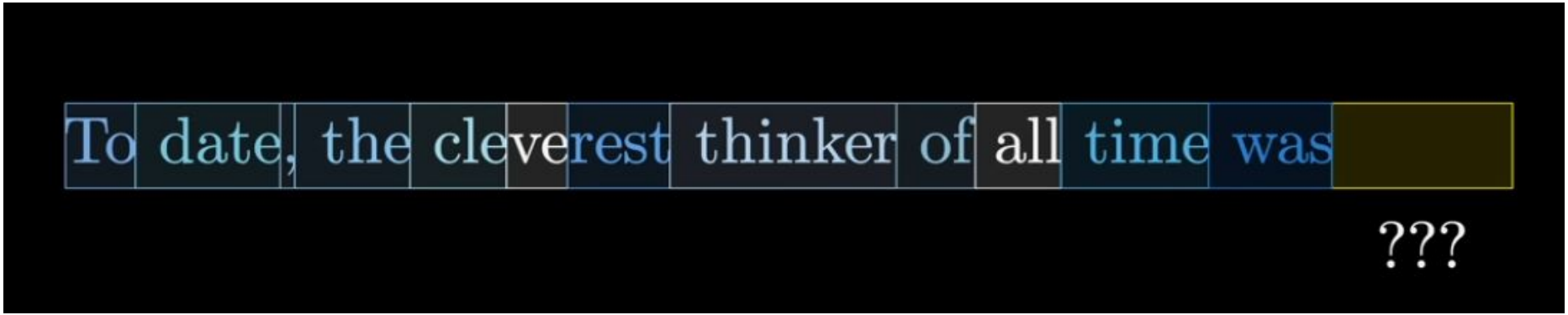
Tokenization - Embedding - Attention - Pre-Training - Transfer Learning

What are Large Language Models (LLMs)?

- The prompt is evaluated in the embedding space to determine what parts are most important to attend to.
- The model can be augmented or specialized for different tasks or domains.
    - This is expensive in every sense of the word.
- The pre-trained set of embeddings can be transferred to perform better for different kinds of specialized prompts.
    - Again, this is similar to the vision parsing networks.

Tokenization - Embedding - Attention - Pre-Training - Transfer Learning

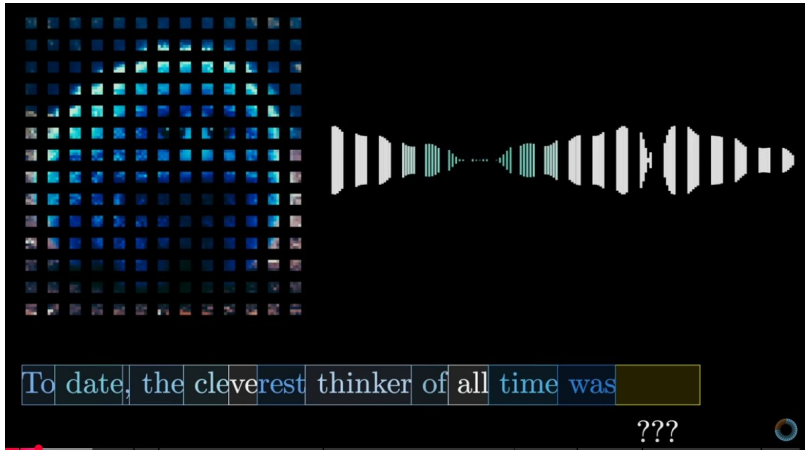
Tokens



https://www.youtube.com/watch?v=wjZofJX0v4M

What are Large Language Models (LLMs)?

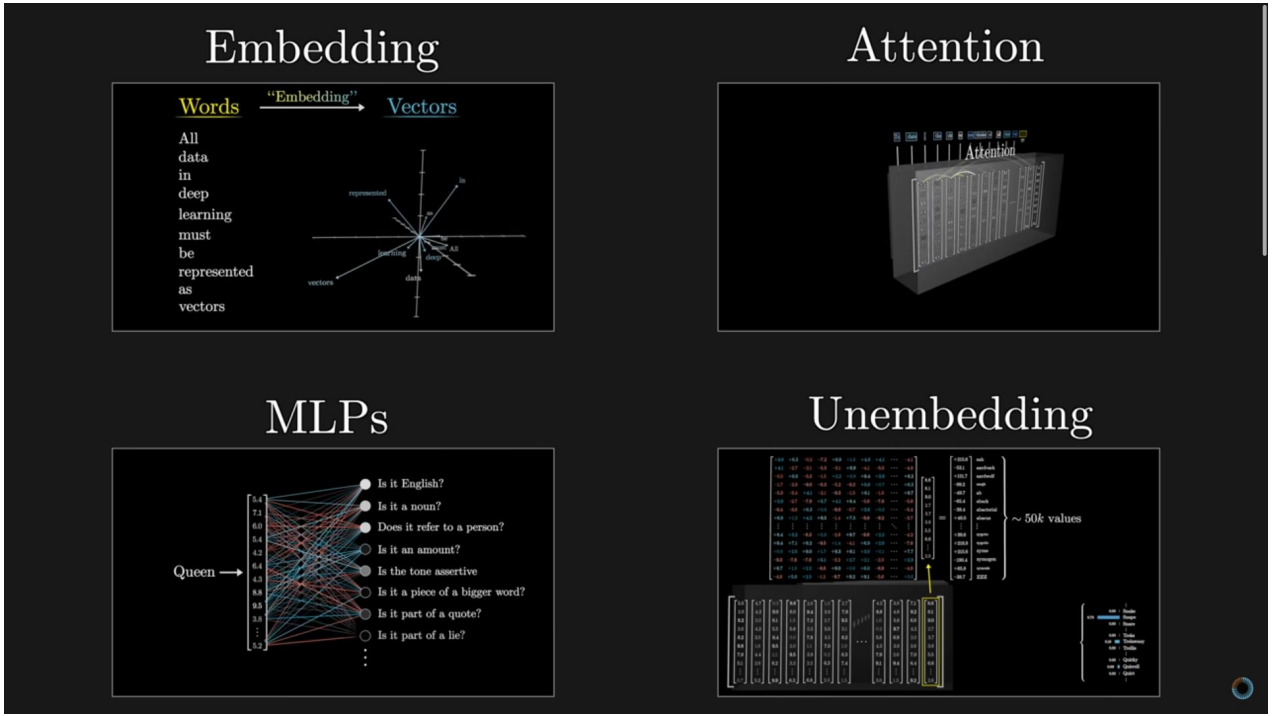
- During model training, all the text is encoded into the model space as tokens.
  - A token can be letters and punctuation, but it can also be whole words or parts of speech.
- The layers of the LLM weight the tokens together into more complex topics.
  - Similar to the layers in vision networks, segmenting image facets of increasing complexity.
- The further apart phrases or topics are in the trained semantic space, the less related they are.



https://julien-vitay.net/lecturenotes-neurocomputing/3-deeplearning/3-CNN.htm; https://www.youtube.com/watch?v=wjZofJX0v4M

https://www.youtube.com/watch?v=wjZofJX0v4M

A Visual Reference - 3Blue1Brown



https://www.youtube.com/watch?v=wjZofJX0v4M

Questions?

https://www.youtube.com/watch?v=5hfYJsQAhl0

The Impact of Large Language Models

How are Large Language Models Used?

- Text Generation (Chatbots)
  - Customer Service
  - Talk with your documents
- Coding assistants
  - Github Copilot
  - Coding teams
- News Reports - Summaries
  - AI generated news stories
  - Academic Work
- Search Engines
  - Google called a code red
  - Databases - Search Replacement?

https://www.forbes.com/sites/bernardmarr/2023/01/10/the-11-best-and-most-innovative-new-gadgets-and-devices-in-2023/

Impact on Neuro- and Data Science

Article | Published: 07 April 2025

Sociodemographic biases in medical decision making by large language models

Mahmud Omar, Shelly Soffer, Reem Agbareia, Nicola Luigi Bragazzi, Donald U. Apakama, Carol R. Horowitz, Alexander W. Charney, Robert Freeman, Benjamin Kummer, Benjamin S. Glicksberg, Girish N. Nadkarni & Eyal Klang

Nature Medicine (2025) | Cite this article

"Similarly, cases labeled as having high-income status received significantly more recommendations ($P < 0.001$) for advanced imaging tests… while low- and middle-income-labeled cases were often limited to basic or no further testing."

Benchmarking large language models for biomedical natural language processing applications and recommendations

Qingyu Chen, Yan Hu, Xueqing Peng, Qianqian Xie, Qiao Jin, Aidan Gilson, Maxwell B. Singer, Xuguang Ai, Po-Ting Lai, Zhizheng Wang, Vipina K. Keloth, Kalpana Raja, Jimin Huang, Huan He, Fongci Lin, Jingcheng Du, Rui Zhang, W. Jim Zheng, Ron A. Adelman, Zhiyong Lu & Hua Xu

Nature Communications **16**, Article number: 3280 (2025) | Cite this article

"Open-source LLMs still require fine-tuning to close performance gaps. We find issues like missing information and hallucinations in LLM outputs. These results offer practical insights for applying LLMs in BioNLP (Biomedical Natural Language Processing)."

Perspective | Open access | Published: 25 January 2025

Retrieval-augmented generation for generative artificial intelligence in health care

Rui Yang, Yilin Ning, Emilia Keppo, Mingxuan Liu, Chuan Hong, Danielle S. Bitterman, Jasmine Chiat Ling Ong, Daniel Shu Wei Ting & Nan Liu

npj Health Systems **2**, Article number: 2 (2025) | Cite this article

"In this perspective, we analyze the possible contributions that RAG could bring to health care in equity, reliability, and personalization. Additionally, we discuss the current limitations and challenges of implementing RAG in medical scenarios."

Article | Open access | Published: 21 January 2025

What large language models know and what people think they know

Mark Steyvers, Heliodoro Tejeda, Aakriti Kumar, Catarina Belem, Sheer Karny, Xinyue Hu, Lukas W. Mayer & Padhraic Smyth

Nature Machine Intelligence **7**, 221–231 (2025) | Cite this article

"The findings indicate that there is a significant gap, as measured by calibration and discrimination, between what LLMs know and what humans believe they know based on default explanations."

And everyone is hyped!

GitHub Copilot

Code 55% Faster!

Revolutionizing Data Annotation: The Pivotal Role of Large Language Models

By **Adnan Hassan** - March 3, 2024

Nvidia CEO predicts the death of coding — Jensen Huang says AI will do the work, so kids don't need to learn

News By Benedict Collins published February 26, 2024

Jensen Huang believes coding languages are a thing of the past

On the Utility of Large Language Model Embeddings for Revolutionizing Semantic Data Harmonization in Alzheimer's and Parkinson's Disease

Yasamin Salimi, Tim Adams, Mehmet Can Ay, Helena Balabin, Marc Jacobs, and 1 more

And everyone is hyped!… right?

GitHub Copilot

"Downward Pressure on Code Quality"

Use of large language models might affect our cognitive skills

Richard Heersmink

Nature Human Behaviour (2024) | Cite this article

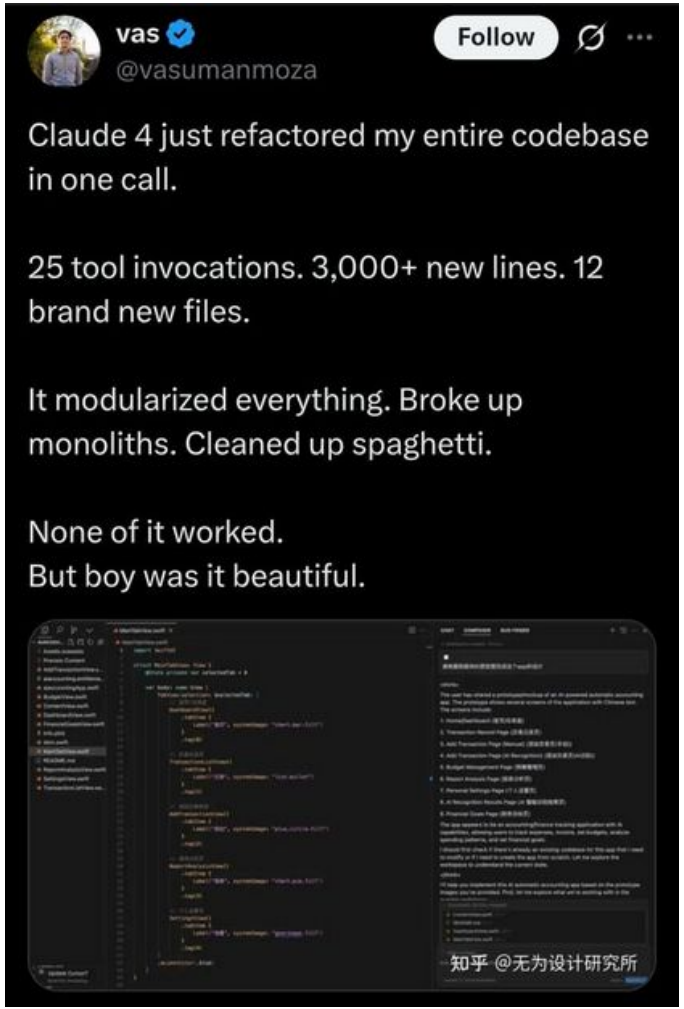
How AI generated code compounds technical debt

"I don't think I have ever seen so much technical debt being created in such a short period of time"

By Bill Doerrfeld

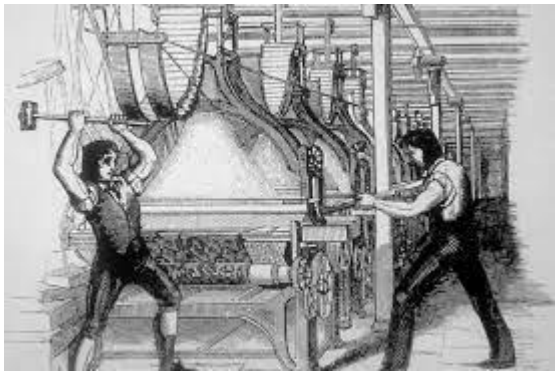
Vibe Coding

- Programming an “app” exclusively with LLM tools.
- You may not even know the programming language you’re working with.
- While this can be fun for pet projects, ***you are responsible for the claims you make with your data science projects***.


vas
@vasumanmoza
Follow
Claude 4 just refactored my entire codebase in one call.
25 tool invocations. 3,000+ new lines. 12 brand new files.
It modularized everything. Broke up monoliths. Cleaned up spaghetti.
None of it worked.
But boy was it beautiful.
知乎 @无为设计研究所

This isn’t the first time…

- The industrial revolution greatly reduced needs of common types of skilled labor.
- A period of upheaval and consolidation, but eventually a new normal.
  - Improved quality of life (arguably).
- However, not everybody was on board.
  - Luddites



https://en.wikipedia.org/wiki/Luddite; https://www.historyextra.com/period/industrial-revolution/who-were-luddites-facts-what-happened/; https://www.linkedin.com/pulse/article-2-steam-steel-strategy-industrial-revolution-thomas-petrou

How do we do this right?

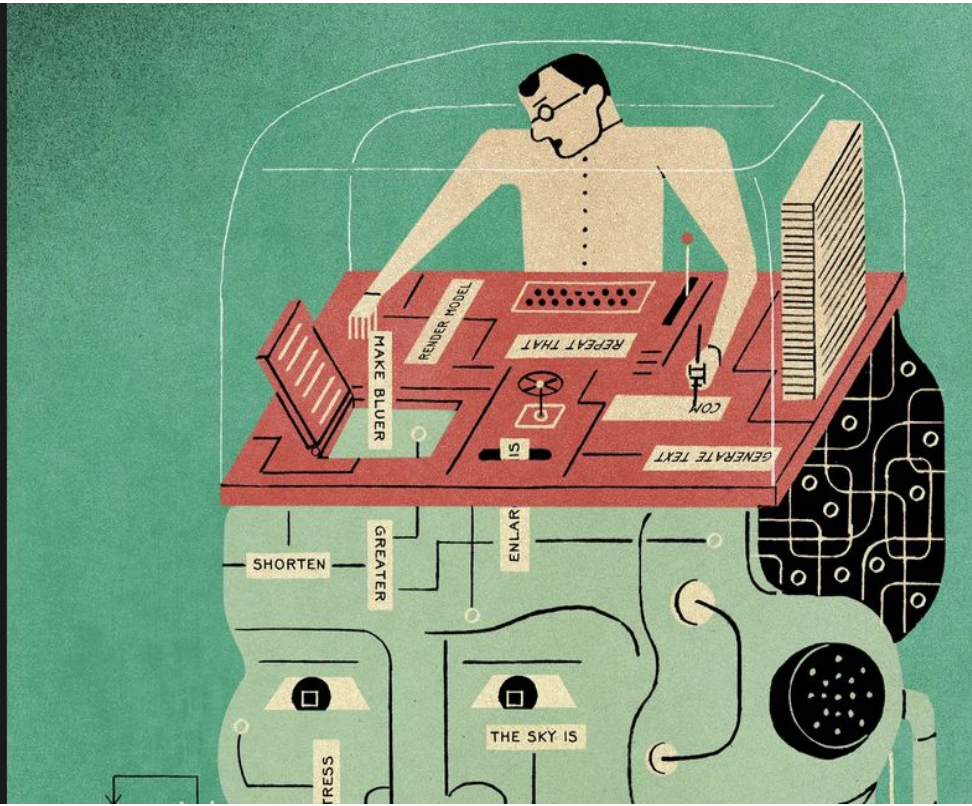
- There is a lot of potential for LLMs to greatly improve our lives.
  - Automate away many more tedious, detail oriented tasks.
  - Provide easier access to information.
- There is a lot of problems that may arise from their unchecked use.
  - They are still very unreliable - hallucinations (word play for **errors** and **mistakes**).
  - Corporate control of the largest models is driven by profit, not benefit.
- ***It is critical that we proactively engage with and understand how to best use this technology so it doesn't cause more harm than good.***

Using an LLM Programmatically

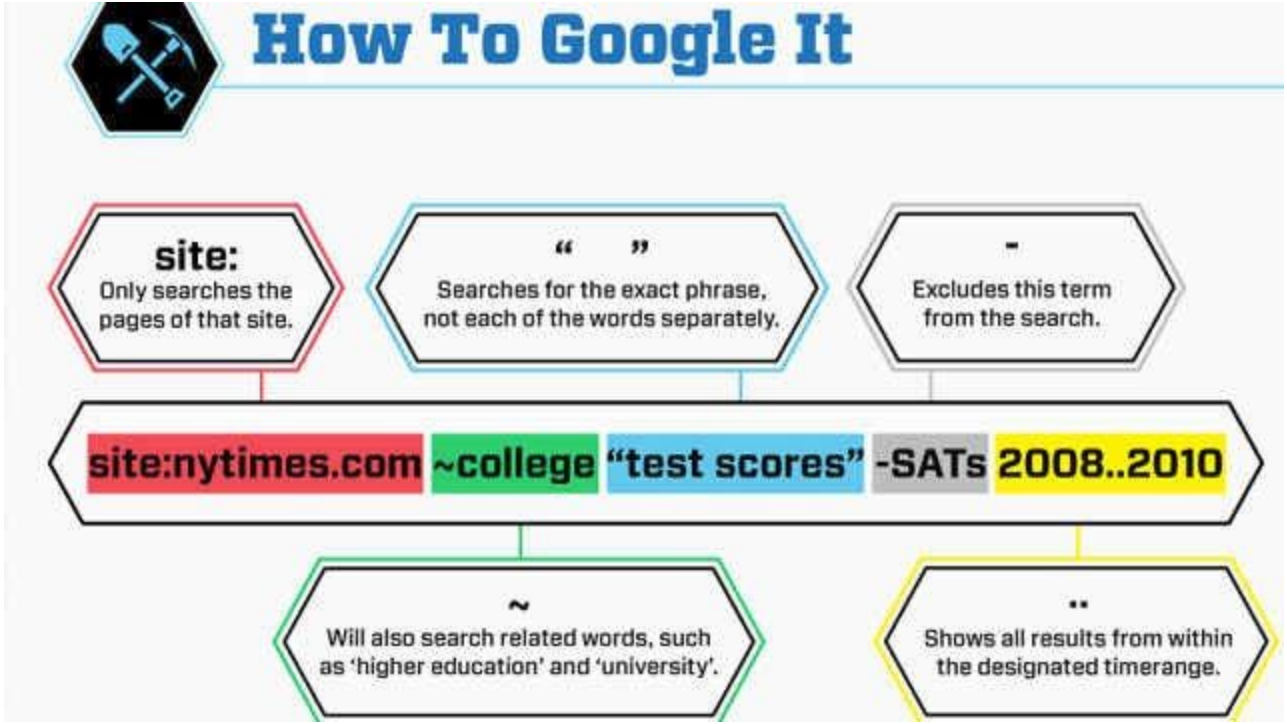
FEATURE | AI

AI PROMPT ENGINEERING IS DEAD

Long live AI prompt engineering



https://towardsdatascience.com/what-i-learned-pushing-prompt-engineering-to-the-limit-c40f0740641f; https://spectrum.ieee.org/prompt-engineering-is-dead


How To Google It
site:
Only searches the pages of that site.
" "
Searches for the exact phrase, not each of the words separately.
-
Excludes this term from the search.
site:nytimes.com ~college "test scores" -SATs 2008..2010
~
Will also search related words, such as 'higher education' and 'university'.
..
Shows all results from within the designated timerange.

Anatomy of a Prompt

- There are 3 speakers who can “talk” in a prompt.
  - The user
  - The model
  - The system
- Prompts are typically passed as JSON that hold the history of the interaction.

```python
import os
from openai import OpenAI

# pull api key from loaded environment variable
client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

# build a helper function w/ new API to return messages
def get_completion(prompt, model="gpt-3.5-turbo"):
    messages = [{"role": "user", "content": prompt}]
    response = client.chat.completions.create(messages=messages, model=model)
    return response.choices[0].message.content

# test the call
get_completion("Why is the sky blue?")
```

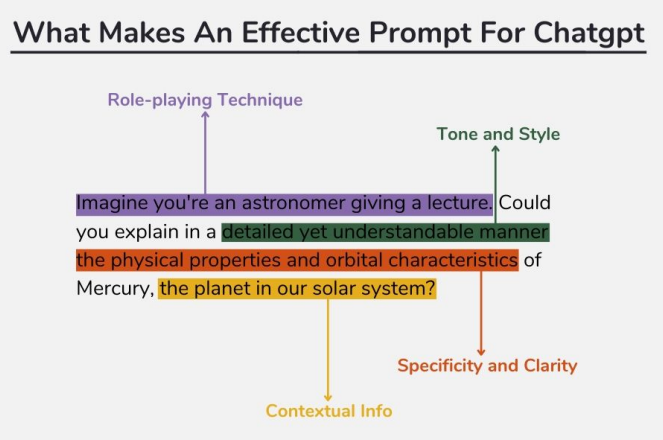
```python
def get_completion_from_messages(messages, model="gpt-3.5-turbo", temp=0):
    response = client.chat.completions.create(model=model,
                                              messages=messages,
                                              temperature=temp)
    return response.choices[0].message.content

messages =  [
{'role':'system', 'content':'You are an assistant that speaks like Shakespeare.'},
{'role':'user', 'content':'tell me a joke'},
{'role':'assistant', 'content':'Why did the chicken cross the road'},
{'role':'user', 'content':'I don\'t know'}  ]

response = get_completion_from_messages(messages, temperature=1)
```

Anatomy of a Prompt

- The **user** is what you (the user) are asking the model to produce.
- The **model** is the LLMs responses to any input.
- The **system** is a silent setting that can modify how the LLM responds.



https://softgist.com/chatgpt-prompt-engineering-effective-prompt

Prompt Engineering - General Guidelines

1. Be clear and specific in what you ask.
   - This does not mean be brief.
2. Use delimiters to separate out specific parts or the prompt.
   - i.e. summarize the text in triple quotes.
   - Helps to sanitize the inputs of the LLMs.
3. Ask for structured output in the response
   - JSON formatting, HTML tables, etc.
4. Ask the LLM to check input conditions as part of the prompt.
5. Provide successful input / output pairs for it to mimic.
6. Give the model time to “think”.
   - Give the LLM instructions for incremental steps toward the desired outcome
   - Have the LLM show its work while hiding its reasoning from the user.
7. Tell the model to make its own solution and compare it to the input
   - Again, step by step helps.

```
text = f"""
You should express what you want a model to do by \
providing instructions that are as clear and \
specific as you can possibly make them. \
This will guide the model towards the desired output, \
and reduce the chances of receiving irrelevant \
or incorrect responses. Don't confuse writing a \
clear prompt with writing a short prompt. \
In many cases, longer prompts provide more clarity \
and context for the model, which can lead to \
more detailed and relevant outputs.
"""

prompt = f"""
Summarize the text delimited by triple backticks \
into a single sentence.
```{text}```
"""
response = get_completion(prompt)
print(response)
```
```

```
prompt = f"""
Generate a list of three made-up book titles along \
with their authors and genres.
Provide them in JSON format with the following keys:
book_id, title, author, genre.
"""
response = get_completion(prompt)
print(response)
```

```
text_1 = f"""
Making a cup of tea is easy! First, you need to get some \
water boiling. While that's happening, \
grab a cup and put a tea bag in it. Once the water is \
hot enough, just pour it over the tea bag. \
Let it sit for a bit so the tea can steep. After a \
few minutes, take out the tea bag. If you \
like, you can add some sugar or milk to taste. \
And that's it! You've got yourself a delicious \
cup of tea to enjoy.
"""
prompt = f"""
You will be provided with text delimited by triple quotes.
If it contains a sequence of instructions, \
re-write those instructions in the following format:

Step 1 - ...
Step 2 - …
…
Step N - …

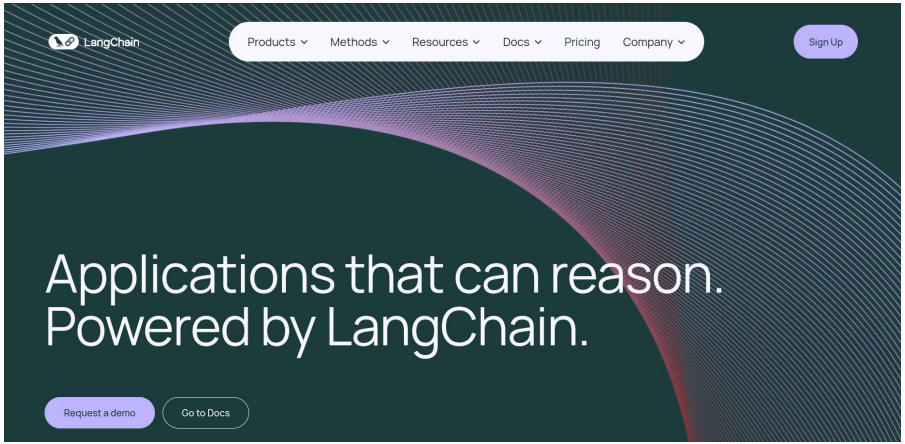
If the text does not contain a sequence of instructions, \
then simply write \"No steps provided.\"

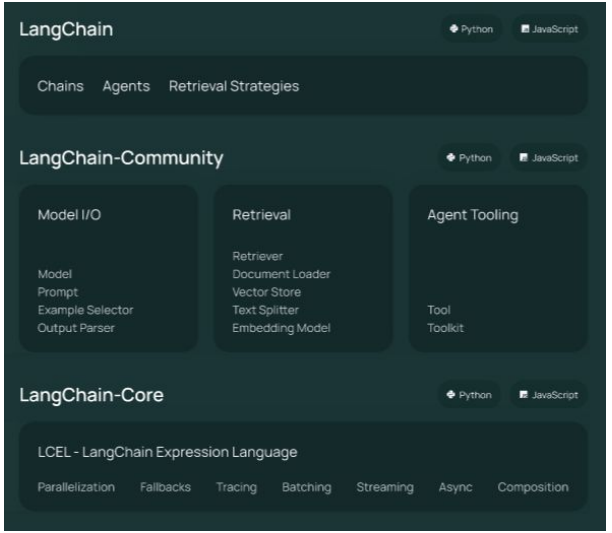
\"\"\"{text_1}\"\"\"
"""
response = get_completion(prompt)
print("Completion for Text 1:")
print(response)
```

Specific Tools

LangChain

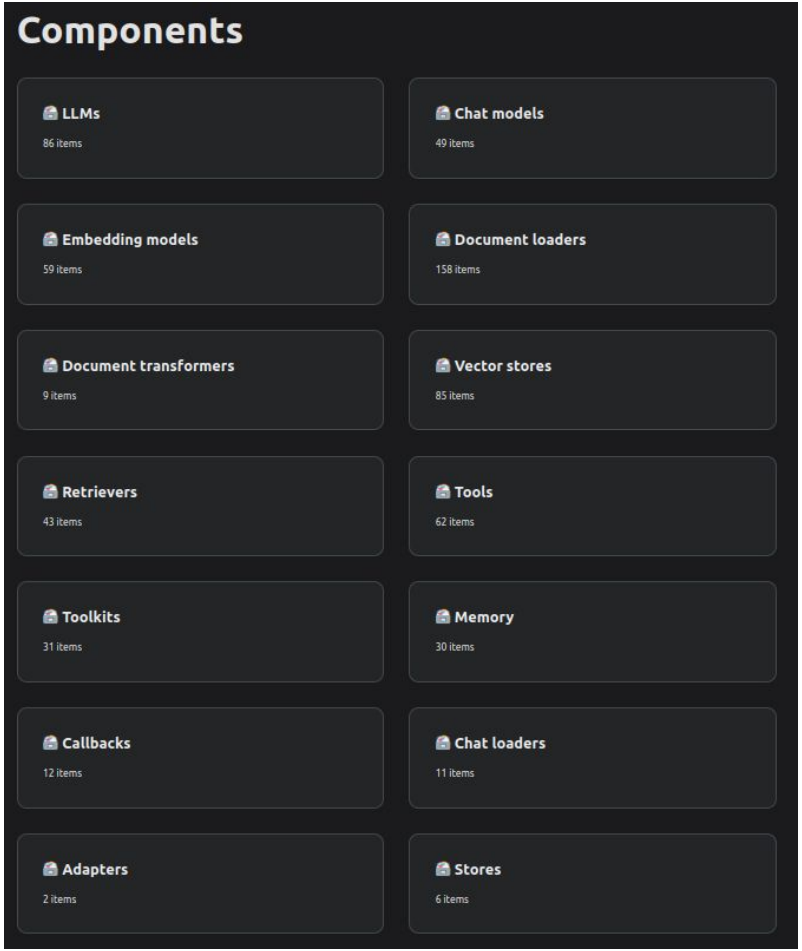
- Python / JS tools for interacting with LLMs programmatically.
- Facilitates prompt engineering and multiple calls to answer a question.
- Many utilities for parsing data for ~~vector embedding in **RAG**s~~ **Agentic Workflows**.
- Allows for (relatively) painless swapping between different LLMs.





LangChain Components

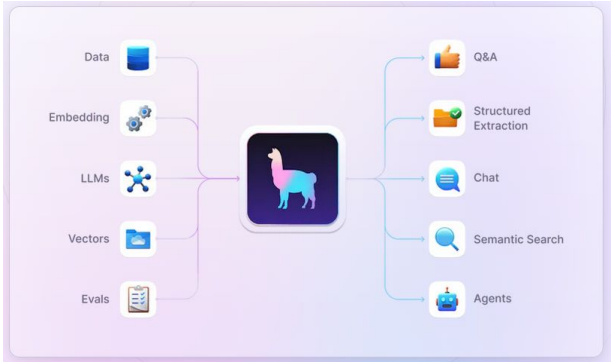
- Classes / Modules for working with standard components.
- Prompt Templates can be created with many standard strategies and logic.
  - LCEL (LangChain Expression Language)
- Because LangChain prompts are modular, they can be easily passed to multiple models.

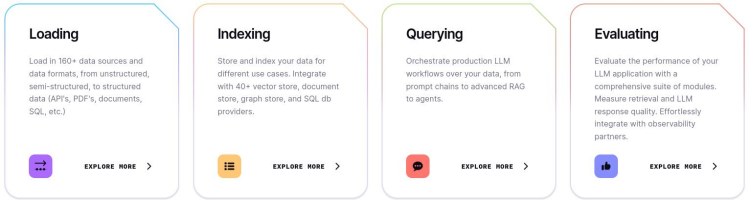


```python
from langchain_core.output_parsers import StrOutputParser
from langchain_core.prompts import ChatPromptTemplate
from langchain_openai import ChatOpenAI

prompt = ChatPromptTemplate.from_template("tell me a short joke about {topic}")
model = ChatOpenAI(model="gpt-4")
output_parser = StrOutputParser()

chain = prompt | model | output_parser

chain.invoke({"topic": "ice cream"})
```

LlamaIndex

- Many overlapping features with LangChain.
- A bit more focus on building Agents with RAGs (Retrieval Augmented Generation).
- These tools are all still rapidly being developed.
    - Some features / models may be further developed in one ecosystem over another.





Sourabh Desai • 2025-05-29

RAG is dead, long live agentic retrieval

LlamaCloud | RAG | Agents | Agentic Document Workflows

RAG has come a long way since the days of naive chunk retrieval; now agentic strategies are table stakes.

https://www.llamaindex.ai/blog/rag-is-dead-long-live-agentic-retrieval

DSPy

Programming—not prompting—Language Models

- Different framework and philosophy.
- Optimize your prompts on an LLM without manually adjusting inputs.
    - Rewrite prompts to optimize a cost function to get closer to your desire. output.

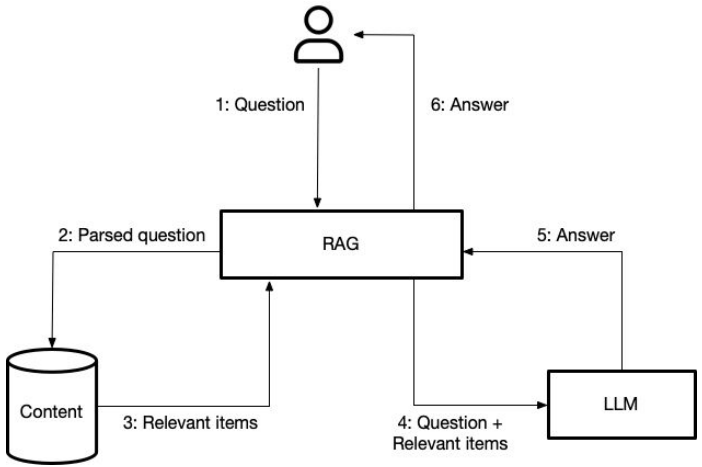
Ollama

- Locally host and run any number of LLMs.
- Integrates with the previously described tools.
- Designed to be similar to running Docker containers.

What ~~are~~ were they all using?

Retrieval Augmented Generation (RAG)

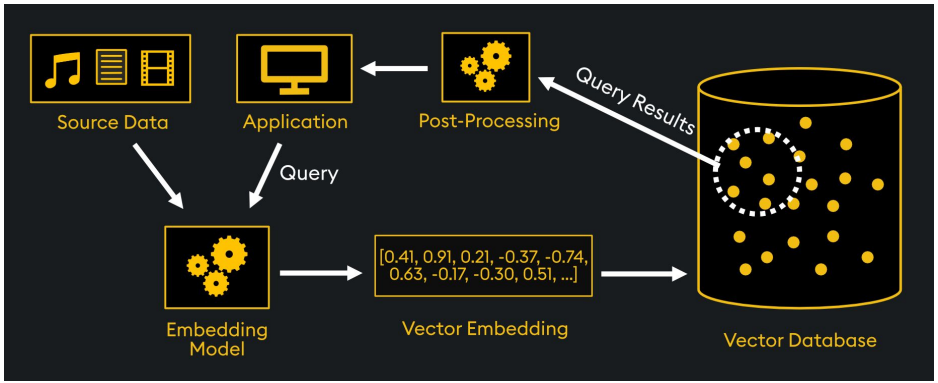
- It is a standard way of augmenting a model to specialize in a task without having to retrain the model.
- This involves a multistep process.
    - LangChain, LlamaIndex, and many other tools can facilitate this.
- The database (vector store) of information is ***very*** important.



```python
from langchain_community.vectorstores import DocArrayInMemorySearch
from langchain_core.output_parsers import StrOutputParser
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.runnables import RunnableParallel, RunnablePassthrough
from langchain_openai.chat_models import ChatOpenAI
from langchain_openai.embeddings import OpenAIEmbeddings

vectorstore = DocArrayInMemorySearch.from_texts(
    ["harrison worked at kensho", "bears like to eat honey"],
    embedding=OpenAIEmbeddings(),
)
retriever = vectorstore.as_retriever()

template = """Answer the question based only on the following context:
{context}

Question: {question}
"""
prompt = ChatPromptTemplate.from_template(template)
model = ChatOpenAI()
output_parser = StrOutputParser()

setup_and_retrieval = RunnableParallel(
    {"context": retriever, "question": RunnablePassthrough()}
)
chain = setup_and_retrieval | prompt | model | output_parser

chain.invoke("where did harrison work?")
```

Vector Databases

- It is a database of information that is embedded in a model's space.
- The prompt is passed to a vector database to identify other relevant information based on the semantic distance between the prompt and the encoded information.
- The identified information is used to augment the prompt and improve the performance of the model.



https://kdb.ai/learning-hub/fundamentals/vector-database-101/

All kinds of data can be used for added context or tooling

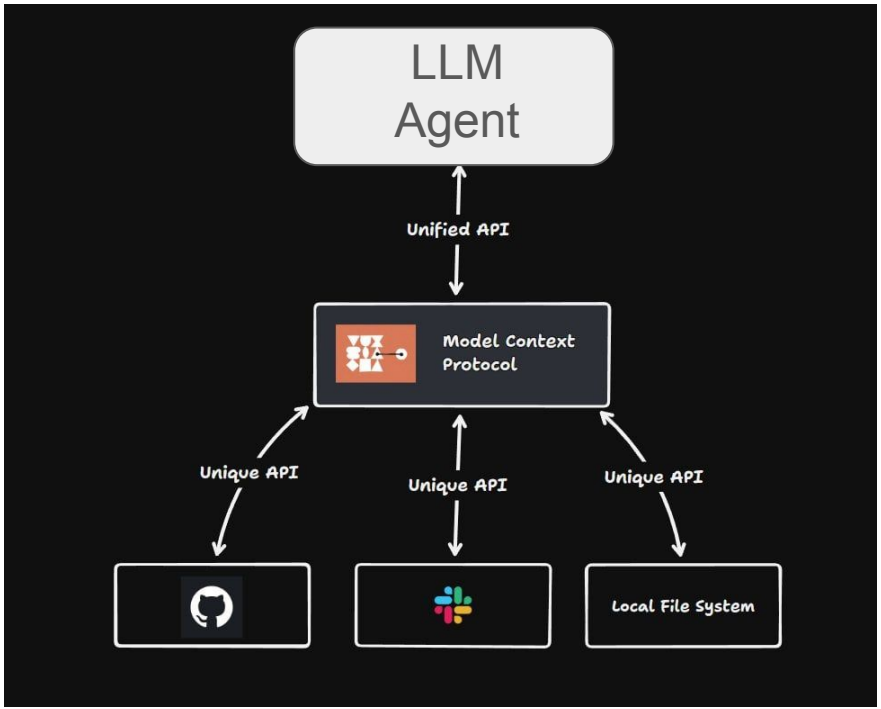
- Pubmed / Arxiv
- Git / Github
- Slack / Gmail
- Project Gutenberg
- Obsidian / Notion
- YouTube transcripts
- Google Services
- AWS
- Microsoft Office Suite
- Most common databases

So what changed?

- The context windows of models got *much* larger.
  - From ~16k tokens to >128k tokens.
  - To “talk” with .pdf article, the entire article can just be passed as context now.
- “Tool” interfaces became much more common and useful.
  - Tools are programs that the LLM can choose to run for external context.
- A standard method to incorporate specific contexts to existing LLMs was necessary.
  - Enter the **M**odel **C**ontext **P**rotocol (**MCP**)

Model Context Protocol (MCP)

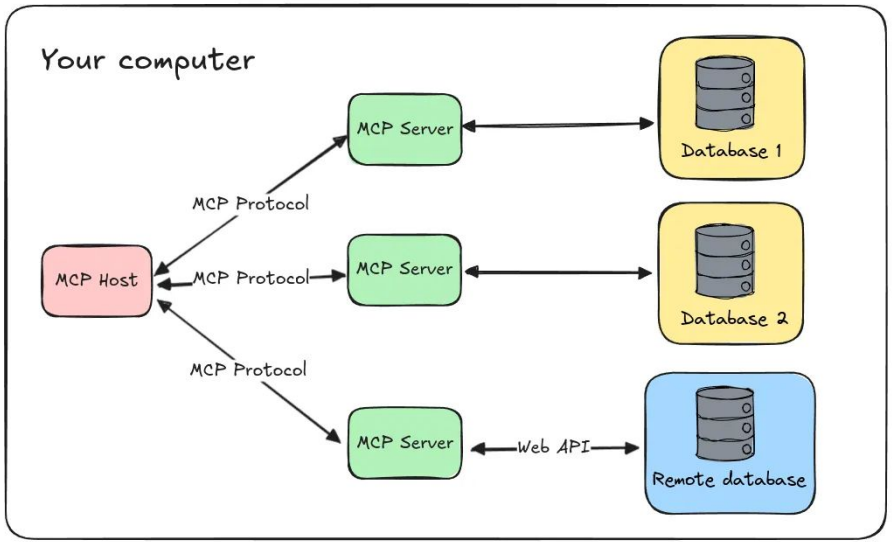
- Introduced by Anthropic, the MCP is a standardized way for LLM applications to communicate with and access external tools, data, and services.
- This is useful for working with secure or private data.
- It keeps the model private.



https://astrix.security/glossary/model-context-protocol-mcp/

Model Context Protocol (MCP)

- An MCP “server” can be deployed in front of a data source.
  - Knowledge Base
  - Web Search
  - Anything that manages data that might be useful for an Agents context.
- By using this standard API, an LLM Agent can selectively access secure, contextual tools for “local” use.



https://medium.com/@sahin.samia/understanding-the-model-context-protocol-mcp-a-comprehensive-explainer-aa0e1809a046

Remember, at any point along the way you could be passing (chaining) your input through another LLM call!

What models should we use?

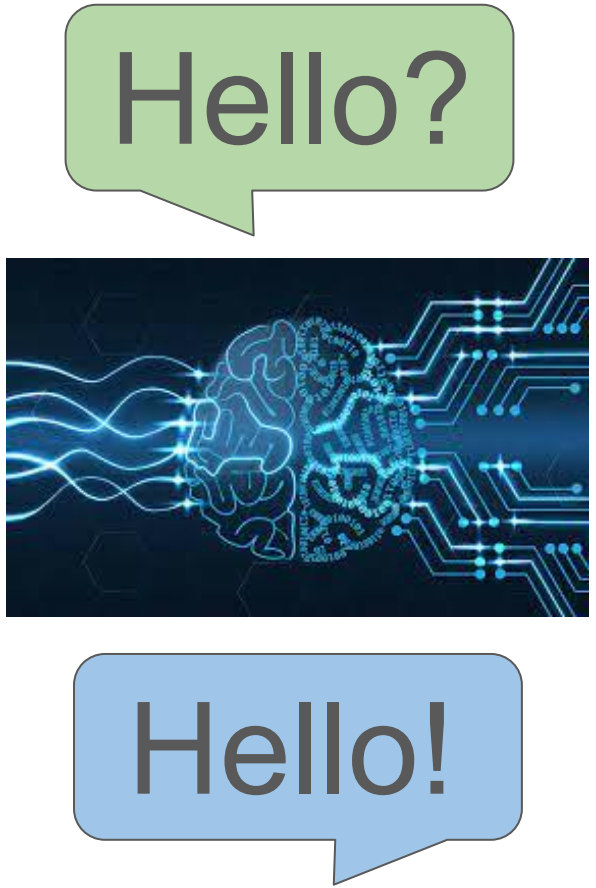
Open Models. Obviously.

- OpenAI is the most popular
  - Its name is also a lie.
- Google Gemini
- DeepSeek
- Meta / Facebook
  - Llama3
  - Open sourced.
  - Many specialized derivatives.
- HuggingFace
  - A full repository of open models and data.

https://the-decoder.com/openllama-is-a-fully-open-source-llm-now-ready-for-business/

Conclusion

- LLMs are a new technology that can fundamentally change how we approach our work.
- It is important to proactively engage and understand their functionality and limits.
- There are many emerging (and open) tools to programmatically work with them.
- Start getting creative solving your problems with LLMs.



https://www.forbes.com/sites/garydrenik/2023/01/11/large-language-models-will-define-artificial-intelligence/

Thanks