

Introduction to the Use of LLMs in Digital Humanities

Quick Starting Guide to Understand Transformer
Architecture, Training Paradigms, Prompting
Techniques, and More

Axel
Pichler

0. Structure

1. Understanding LLMs
 1. Transformer Architecture
 2. Evaluation
2. Application Paradigms
3. Prompt Engineering
4. Potentials and Limitations of the use of LLMs in Digital Humanities
5. Getting Started with LLMs

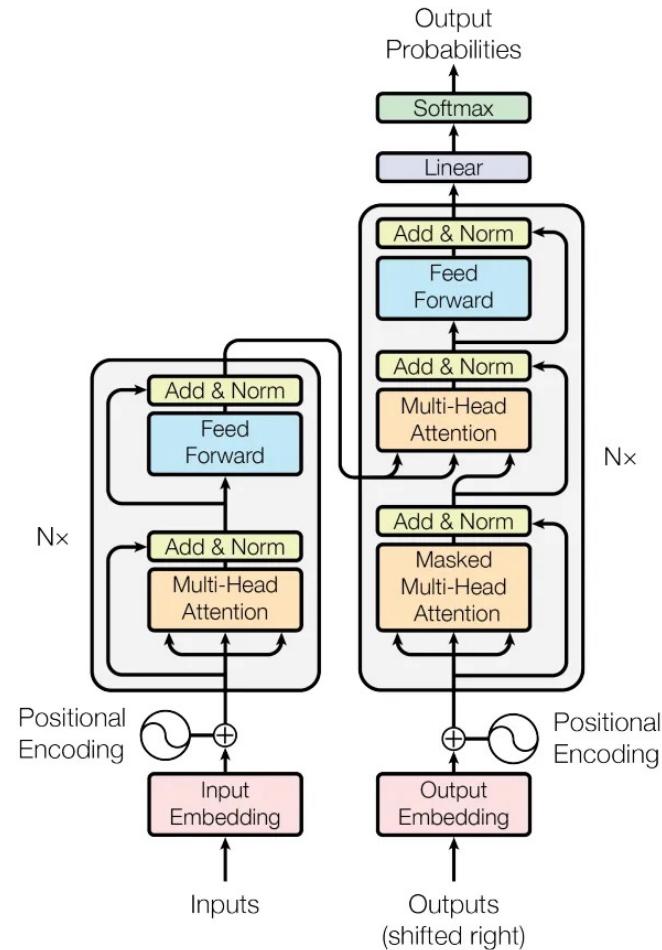
1. Understanding LLMs

1. Understanding LLMs: Transformer Architecture

Vaswani et al. 2017: Attention is all you need

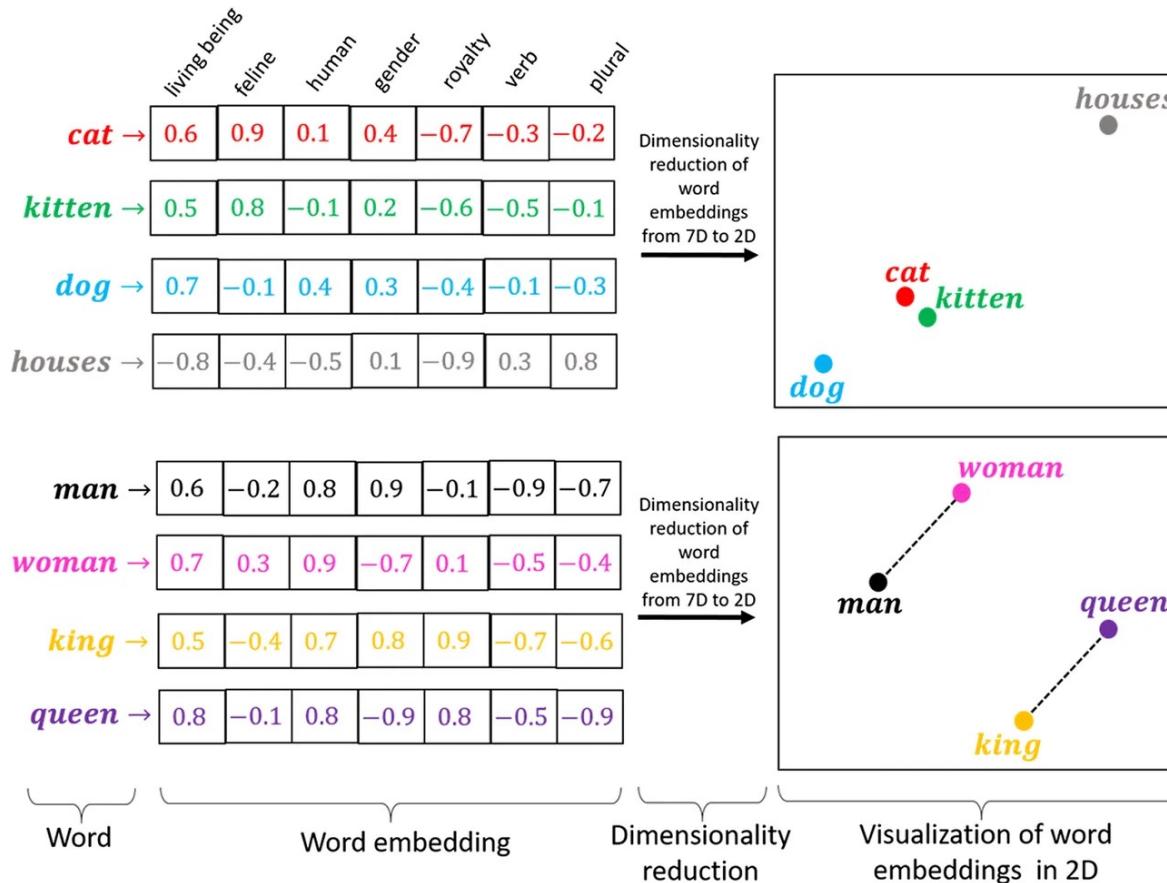
Encoder-Decoder architecture

- Input: Tokens
- Embedding Layer + Positional Encoding
- Encoder Block
 - Multi-Head Self-Attention
 - Add and Norm:
 - Residual Learning
 - Layer Normalization
- Decoder Block



1. Understanding LLMs: Transformer Architecture

Tokenization + Word Embeddings



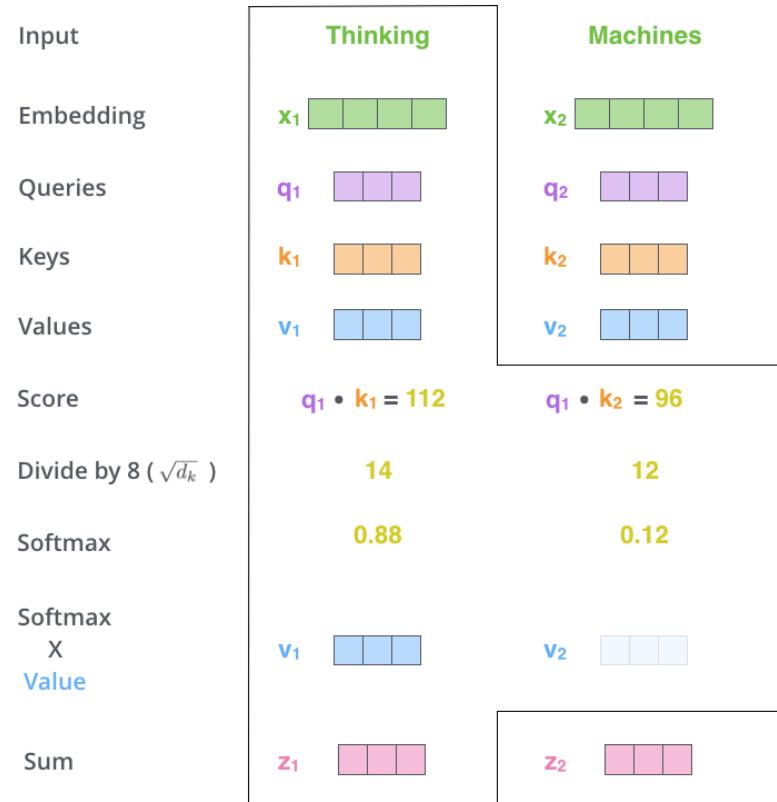
Source: <https://medium.com/@hari4om/word-embedding-d816f643140>

1. Understanding LLMs: Transformer Architecture

Self-Attention: How does it work?

- Input vectors
- Creating Queries, Keys, and Values
- Calculating Attention Scores
- Applying Softmax
- Weighted Sum of Values
- Output Representation

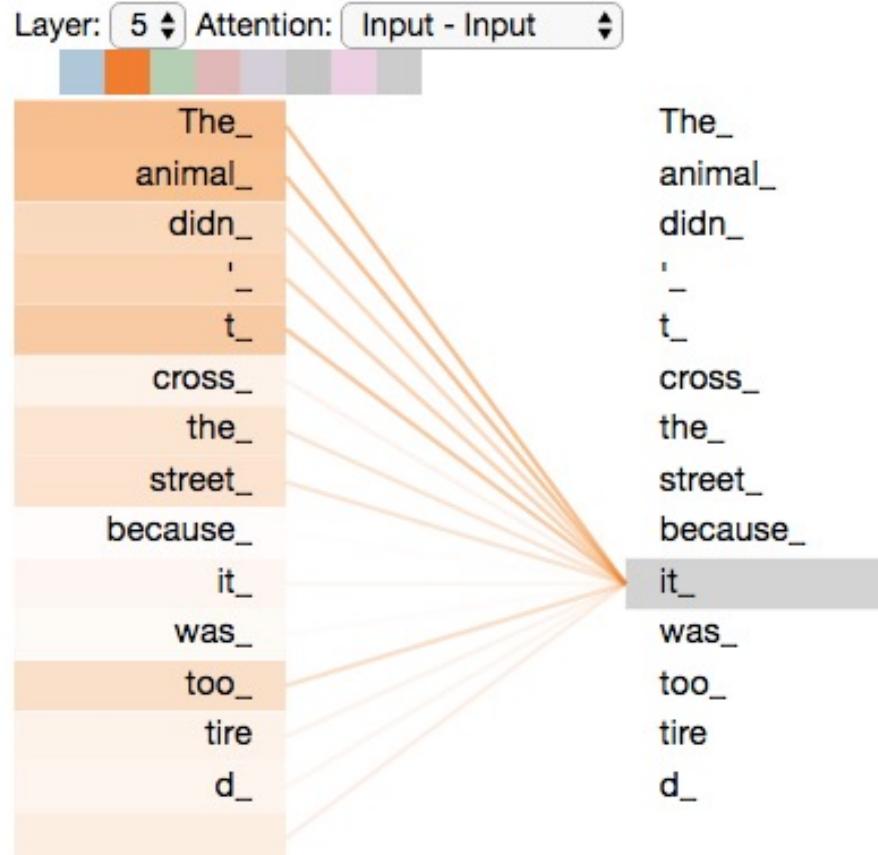
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Source: <https://jalammar.github.io/illustrated-transformer/>

1. Understanding LLMs: Transformer Architecture

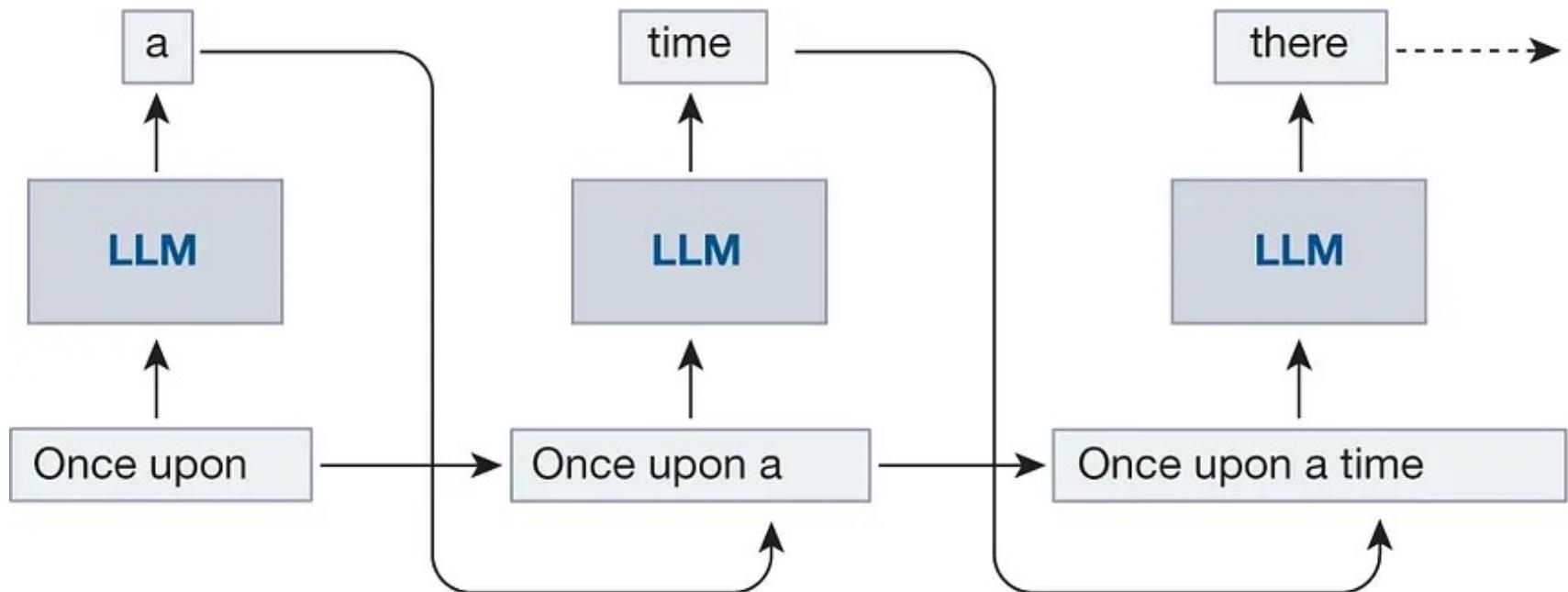
Self-Attention



Source: <https://jalammar.github.io/illustrated-transformer/>

1. Understanding LLMs: Transformer Architecture

Methods to predict the next token: Autoregressive Model



Source: <https://medium.com/@javaid.nabi/all-you-need-to-know-about-lm-text-generation-03b138e0ed19>

1. Understanding LLMs: Transformer Architecture

Methods to predict the next token: Decoding Methods

Deterministic Methods:

- Greedy Search

Stochastic Methods:

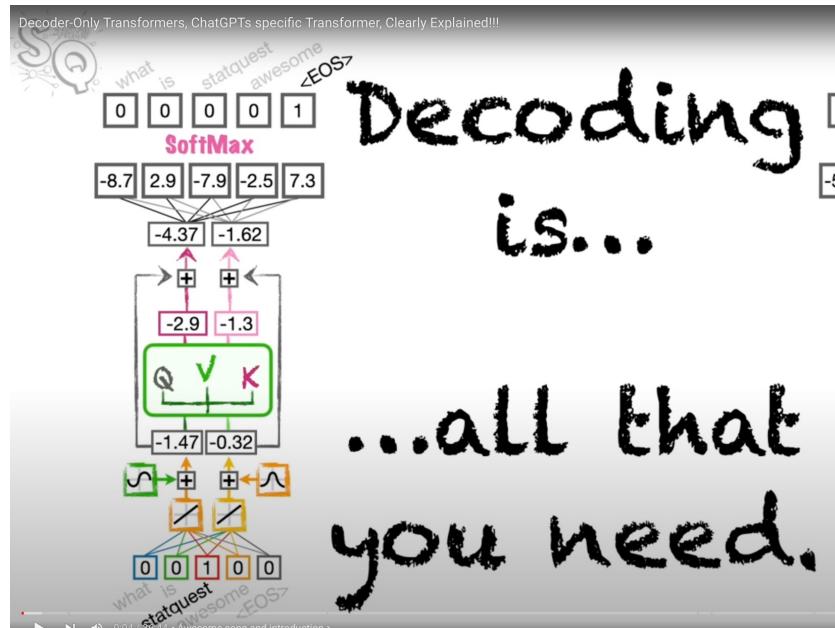
- Temperature
- Top-K Sampling
- Top-p Sampling

```
def get_completion_ant(prompt, temperature=0.7, top_p=1.0, top_k=50):  
    message = client.messages.create(  
        model="claude-3-5-sonnet-20240620",  
        max_tokens=1024,  
        messages=[  
            {"role": "user", "content": prompt}  
        ],  
        temperature=temperature,  
        top_p=top_p,  
        top_k=top_k  
    )  
    return message.content[0].text
```

1. Understanding LLMs: Transformer Architecture

Resources

StatQuest: Decoder-Only Transformers



<https://www.youtube.com/watch?v=bQ5BoolX9Aq>

*Stanford XCS224U: NLU I
Contextual Word Representations*



1. Understanding LLMs: Evaluation

LLM Benchmarks

Open LLM 2 Leaderboard – Benchmarks

- MMLU-Pro (Massive Multitask Language Understanding - Pro version)
- GPQA (Multistep Soft Reasoning)
- MuSR (Multistep Soft Reasoning)
- MATH (Mathematics Aptitude Test of Heuristics, Level 5 subset)
- IFEval (Instruction Following Evaluation)
- BBH (Big Bench Hard)

<https://huggingface.co/spaces/open-llm-leaderboard/blog>

1. Understanding LLMs: Evaluation

https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard

😊 Open LLM Leaderboard

Feeling lost? Documentation is [here](#) 📄

You'll notably find explanations on the evaluations we are using, reproducibility guidelines, best practices on how to submit a model, and our FAQ.

The screenshot shows the Open LLM Leaderboard interface. At the top, there are three buttons: 'LLM Benchmark' (with a trophy icon), 'Submit' (with a rocket icon), and 'Model Vote' (with a upvote icon). Below these are sections for 'Search' (with a text input field) and 'Model types' (with checkboxes for chat models, fine-tuned datasets, pretrained models, and base merges). There are also sections for 'Select Columns to Display' (listing various evaluation metrics like Average, IFEval, BBH, etc.) and 'Precision' (checkboxes for bfloat16 and float16). A slider allows selecting the number of parameters (B) from 6 to 10. At the bottom, there are checkboxes for 'Hide models' (Deleted/incomplete, Merge/MoErge, MoE, Flagged, Show only maintainer's highlight).

T	Model	Average 📈	IFEval	BBH	MATH Lvl 5	GPQA	MUSR	MMLU-PRO
💬	Qwen/Qwen2-72B-Instruct 📄	43.02	79.89	57.48	35.12	16.33	17.17	48.92
💬	meta-llama/Meta-Llama-3-70B-Instruct 📄	36.67	80.99	50.19	23.34	4.92	10.92	46.74
◆	Qwen/Qwen2-72B 📄	35.59	38.24	51.86	29.15	19.24	19.73	52.56
💬	mistralai/Mixtral-8x22B-Instruct-v0.1 📄	34.35	71.84	44.11	18.73	16.44	13.49	38.7
💬	HuggingFaceH4/zephyr-orpo-141b-A35b-v0.1 📄	34.23	65.11	47.5	18.35	17.11	14.72	39.85
💬	microsoft/Phi-3-medium-4k-instruct 📄	33.12	64.23	49.38	16.99	11.52	13.05	40.84
💬	01-ai/Yi-1.5-34B-Chat 📄	33.08	60.67	44.26	23.34	15.32	13.06	39.12
💬	CohereForAI/c4ai-command-r-plus 📄	31.3	76.64	39.92	7.55	7.38	20.42	33.24

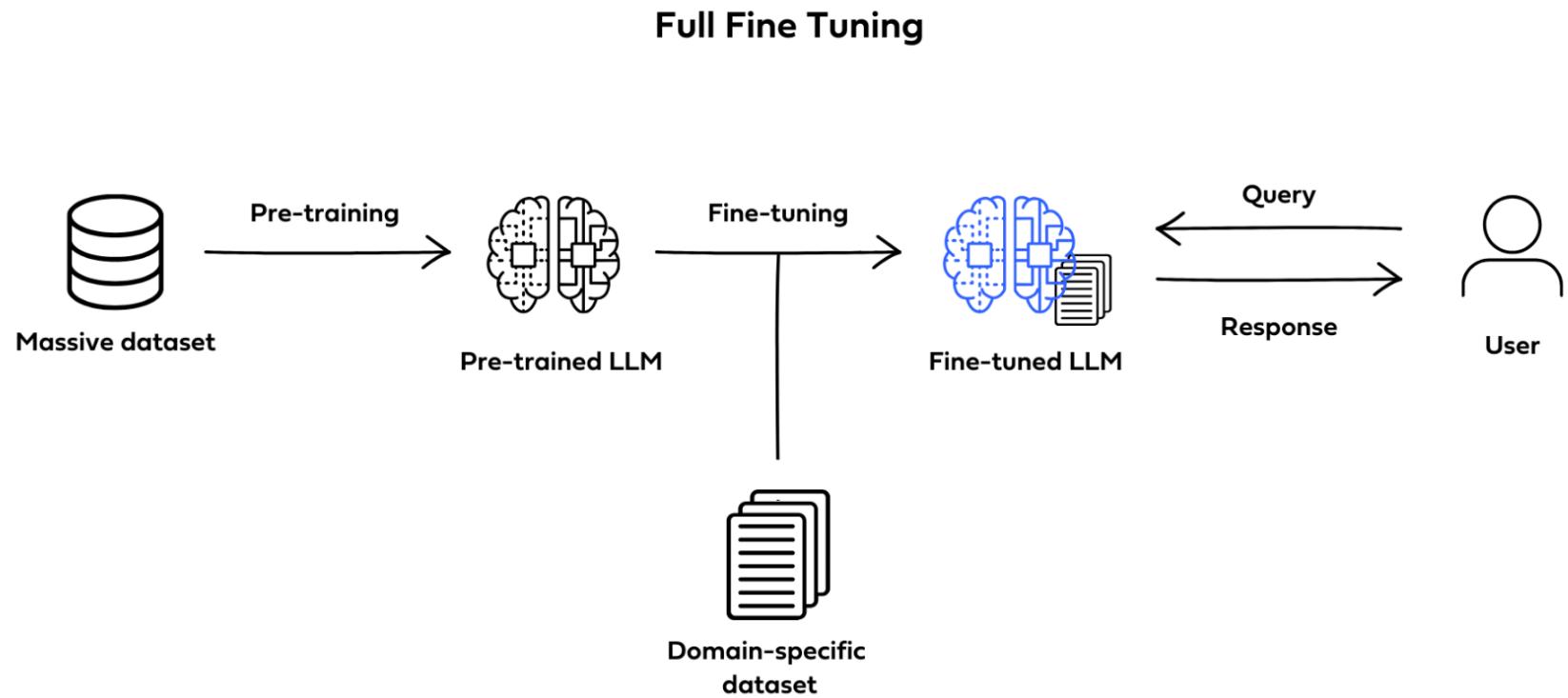
1. Understanding LLMs: Evaluation

Lmsys Leaderboard: <https://arena.lmsys.org>

Rank* (UB)	Model	Arena Elo	95% CI	Votes	Organization	License	Knowledge Cutoff
1	GPT-4o-2024-05-13	1287	+3/-3	42046	OpenAI	Proprietary	2023/10
2	Gemini-Advanced-0514	1267	+3/-4	34382	Google	Proprietary	Online
2	Gemini-1.5-Pro-API-0514	1265	+4/-3	35184	Google	Proprietary	2023/11
4	Gemini-1.5-Pro-API-0409-Preview	1258	+3/-3	55731	Google	Proprietary	2023/11
4	GPT-4-Turbo-2024-04-09	1257	+3/-2	65760	OpenAI	Proprietary	2023/12
6	GPT-4-1106-preview	1251	+2/-3	83330	OpenAI	Proprietary	2023/4
6	Claude 3 Opus	1249	+2/-2	133262	Anthropic	Proprietary	2023/8
6	GPT-4-0125-preview	1246	+3/-2	76586	OpenAI	Proprietary	2023/12
9	Yi-Large-preview	1240	+3/-3	41051	01 AI	Proprietary	Unknown
10	Gemini-1.5-Flash-API-0514	1230	+3/-4	33524	Google	Proprietary	2023/11

2. Application Paradigms

2. Application Paradigms: Pre-Train + Fine-Tune



Source: <https://deci.ai/blog/fine-tuning-peft-prompt-engineering-and-rag-which-one-is-right-for-you/>

2. Application Paradigms

Brown et. al. (2020): Large Language Models are Few-Shot-Learners

In-Context Learning: A »Model is conditioned on a natural language instruction and/or a few demonstrations of the task and is then expected to complete further instances of the task simply by predicting what comes next« (Brown et al. 2020).

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan[†] Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

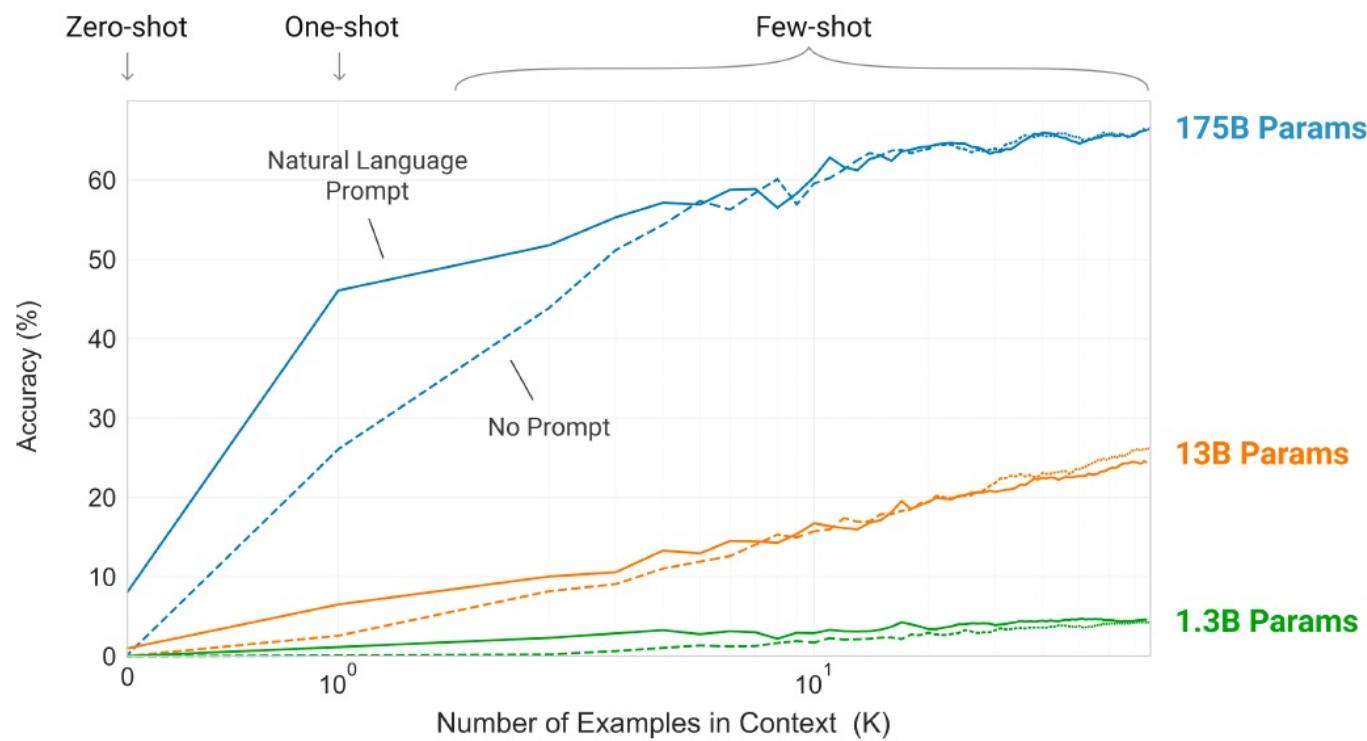
Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

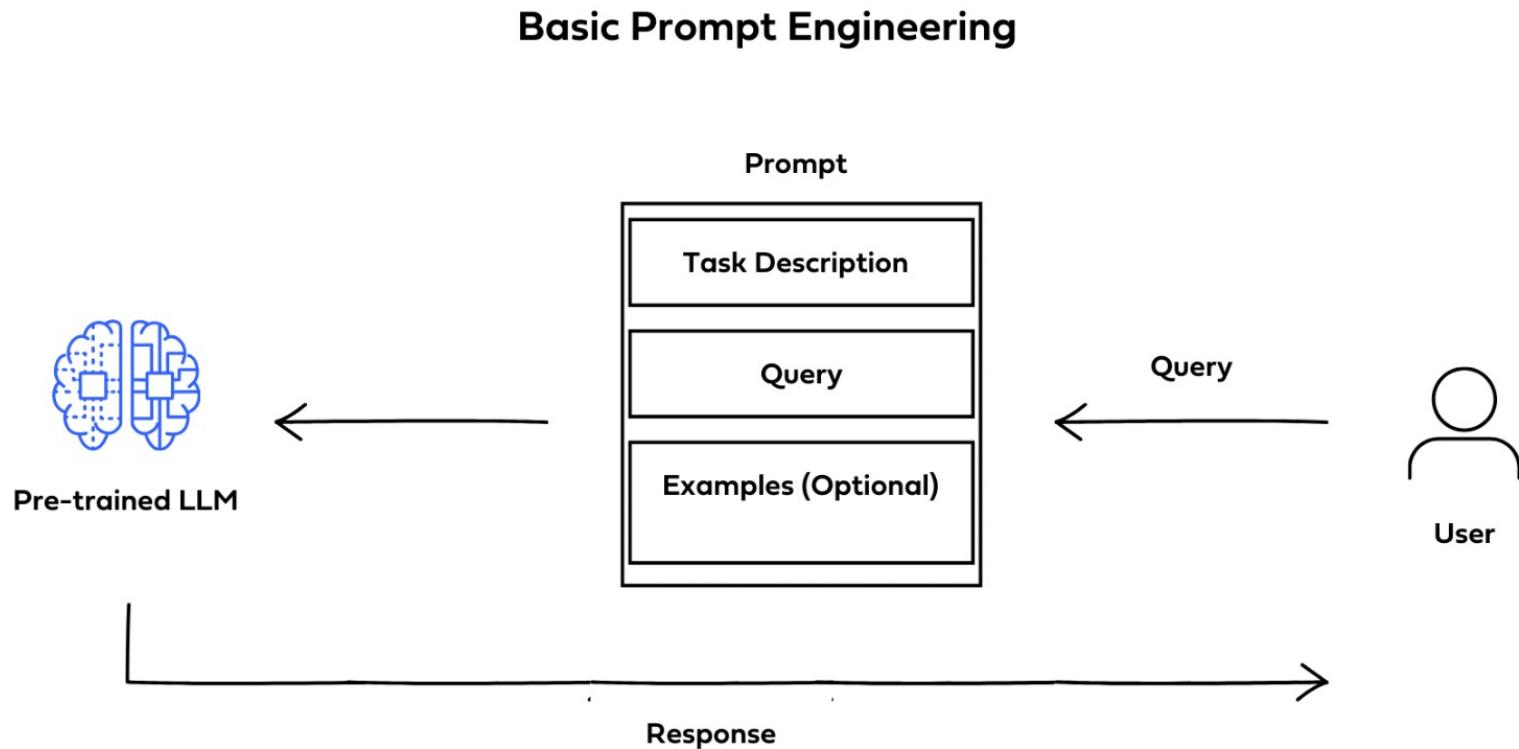
OpenAI

2. Application Paradigms

Brown et. al. (2020): Large Language Models are Few-Shot-Learners



2. Application Paradigms: Pre-Train + Prompt



Source: <https://deci.ai/blog/fine-tuning-peft-prompt-engineering-and-rag-which-one-is-right-for-you/>

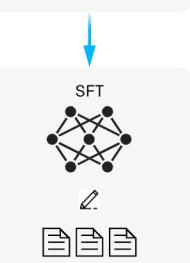
2. Application Paradigms: Pre-Train + Prompt

Aligning language models to follow instructions

Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.



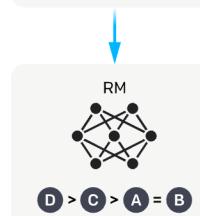
A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.

Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.



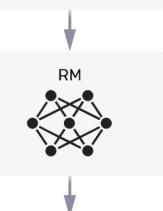
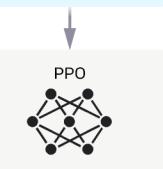
A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.



The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

Source: <https://openai.com/index/instruction-following/>

3. Prompt Engineering

Prompt Engineering:
»[S]trategically designing task-specific instructions, referred to as prompts, to guide model output without altering parameters«

Shahoo et al. 2024

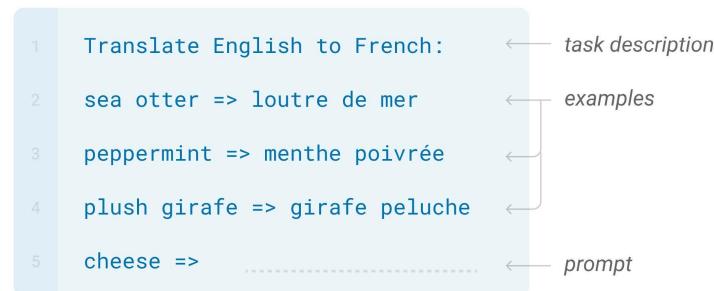
“ ”

3. Prompt Engineering

In-Context Learning

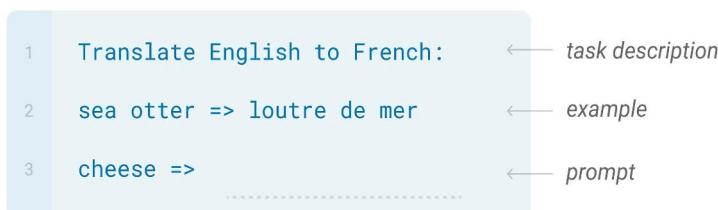
Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



Templates

Few-shot

<question/task and answer pairs>

Q. <question>

A.

One-shot

<question/task and answer pair>

Q. <question>

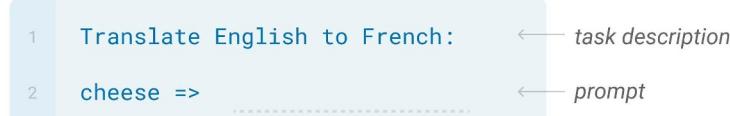
A.

3. Prompt Engineering

In-Context Learning / Chain-of-Thought

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



Templates

Zero-shot

<task description>

A.

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✓

Chain-of-Thought

<question/task>

Let's think step by step

A.

3. Prompt Engineering

Strategies of Engineering Instruction Following Prompts

OpenAI's Prompt Engineering Guide:

- Write clear instructions
 - Ask the model to adapt a persona
 - Use delimiters to clearly indicate distinct parts of the input
 - Specify the steps required to complete a task
 - Provide examples
 - Specify the desired length of the output
- Provide reference text
- Split complex tasks into simpler subtasks
- Give the model time to >think<

3. Prompt Engineering

Strategies of Engineering Instruction Following Prompts

```
prompt=f"""
### Role
You are a medieval scholar with deep knowledge of European history from the 5th to the 15th century.

### Task Description
You will classify excerpts from medieval texts into predefined categories
such as "Historical Events," "Literature," "Religious Texts," and "Biographical Accounts."

### Steps to Complete the Task ###
1. Read the provided text excerpt carefully.
2. Identify the main theme or subject of the excerpt.
3. Choose the most appropriate category from the given list.
4. Justify your classification with a brief explanation.

### Example ###
- **Text Excerpt**: "In the year of our Lord 1066, Duke William of Normandy conquered England, becoming King William I."
  - **Category**: Historical Events
  - **Justification**: The excerpt describes a specific event in history, the Norman Conquest of England.

### Desired Output Length ###
Your classification and justification should be concise, with a maximum of 3 sentences per excerpt.

### Input ###
Medieval text: {text}
"""
```

3. Prompt Engineering

Strategies of Engineering Instruction Following Prompts

Example (Anthropic's Claude-Sonnet-3.5)

```
prompt_topoi = f"""
### Role
You are a medievalist.

### Instruction
Your task is to determine which of the topoi listed below realizes the following Middle High German text from an Arthurian novel.

### Topoi
- courtly world
- forest
- path

### Output format
Answer in one sentence using the following format:
The piece of text realizes the topos ***topos*** because str

"""

Text piece: {snippet_mhd}
Answer: The piece of text realizes the topos
"""

response = get_completion_ant(prompt_topoi)

print(response)
```

The piece of text realizes the topos ***courtly world*** because it depicts a king speaking about his brother and his royal household, using formal language and discussing matters of inheritance and social status typical of medieval courtly settings.

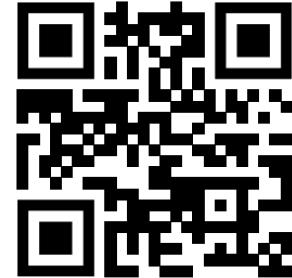
3. Prompt Engineering

Resources

- Prompt Engineering Guide
 - <https://www.promptingguide.ai/en>
- OpenAI Guide:
 - <https://platform.openai.com/docs/guides/prompt-engineering>
- Mistral AI Guide:
 - https://docs.mistral.ai/guides/prompting_capabilities/
- Anthropic Guide:
 - <https://docs.anthropic.com/en/docs/build-with-claude/prompt-engineering/overview>

3. Prompt Engineering

Chatbot Arena: <https://chat.lmsys.org>



A screenshot of the LMSYS Chatbot Arena website interface. At the top, there are several navigation links: "Arena (battle)", "Arena (side-by-side)", "Direct Chat", "Leaderboard", and "About Us". Below these, the main title "LMSYS Chatbot Arena (Multimodal): Benchmarking LLMs and VLMs in the Wild" is displayed. Underneath the title, there are links to "Blog", "GitHub", "Paper", "Dataset", "Twitter", "Discord", and "Kaggle Competition". A section titled "Rules" features a small icon of a book and a list of guidelines. Another section titled "Chatbot Arena Leaderboard" shows a list of models. At the bottom, there is a search bar with the placeholder "Expand to see the descriptions of 70 models" and two model cards labeled "Model A" and "Model B".

- Ask any question to two anonymous models (e.g., ChatGPT, Gemini, Claude, Llama) and vote for the better one!
- You can continue chatting until you identify a winner.
- Vote won't be counted if model identity is revealed during conversation.
- NEW Image Support: **Upload an image** on your first turn to unlock the multimodal arena! Images should be less than 15MB.

Chatbot Arena Leaderboard

- We've collected 1,000,000+ human votes to compute an LLM Elo leaderboard for 100+ models. Find out who is the LLM Champion [here](#)!

Chat now!

Expand to see the descriptions of 70 models

Model A Model B

4. Potentials and Limitations of LLMs in DH

4. Potentials and Limitations of LLMs in DH

Text Classification and Analysis

Classification:

- assign pre-defined classes to objects (e.g., genre to text)
- structured model output (i.e., list or dict objects)

Generation:

- next token prediction

Generative models are primarily not classification models (detour!)

Advantage of LLMs:

- No need for fine-tuning based on several thousand examples

4. Potentials and Limitations of LLMs in DH

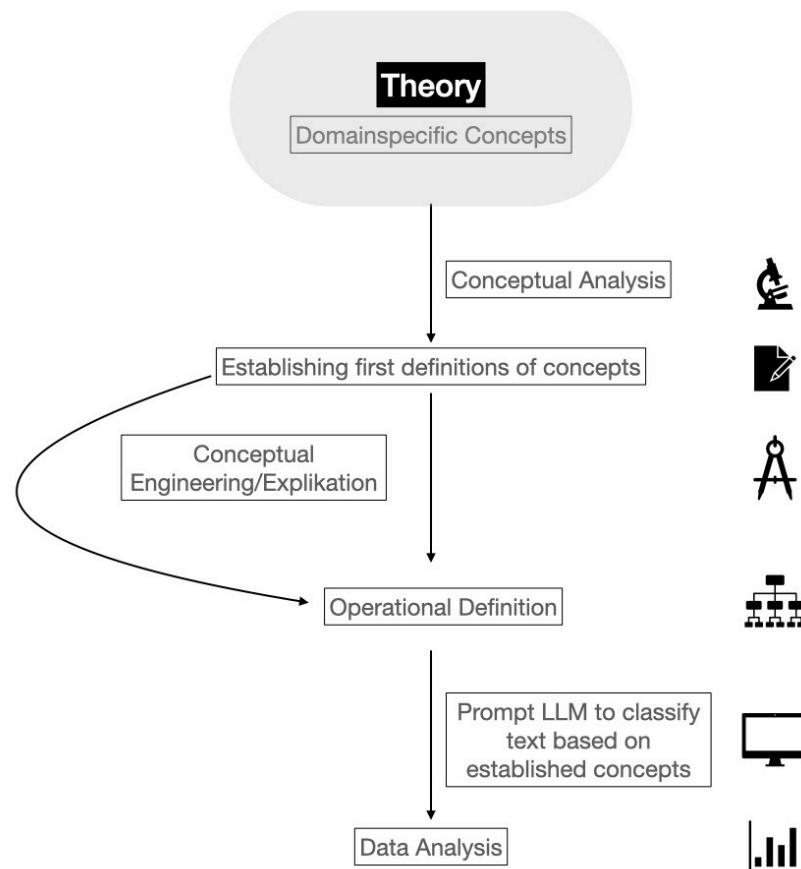
Text Classification and Analysis

Assumptions:

- Candidate units are clearly identifiable and we can refer to them
- Category labels align with everyday language
 - Different observations whether giving definitions in the prompt helps
- Category labels are present in the model's vocabulary

4. Potentials and Limitations of LLMs in DH

Text Classification and Analysis: Workflow



4. Potentials and Limitations of LLMs in DH

Text Classification and Analysis

Options for implementation:

- Different prompting techniques
 - zero-shot, few-shot, instruction-following
- Different tasks
 - Extraction task: List all siblings in the given text.
 - Decision-making task: Is {NE_1} a sibling of {NE_2}?
 - Entailment: Does {sen_1} entail {sen_2}?
- Different ways to express the task

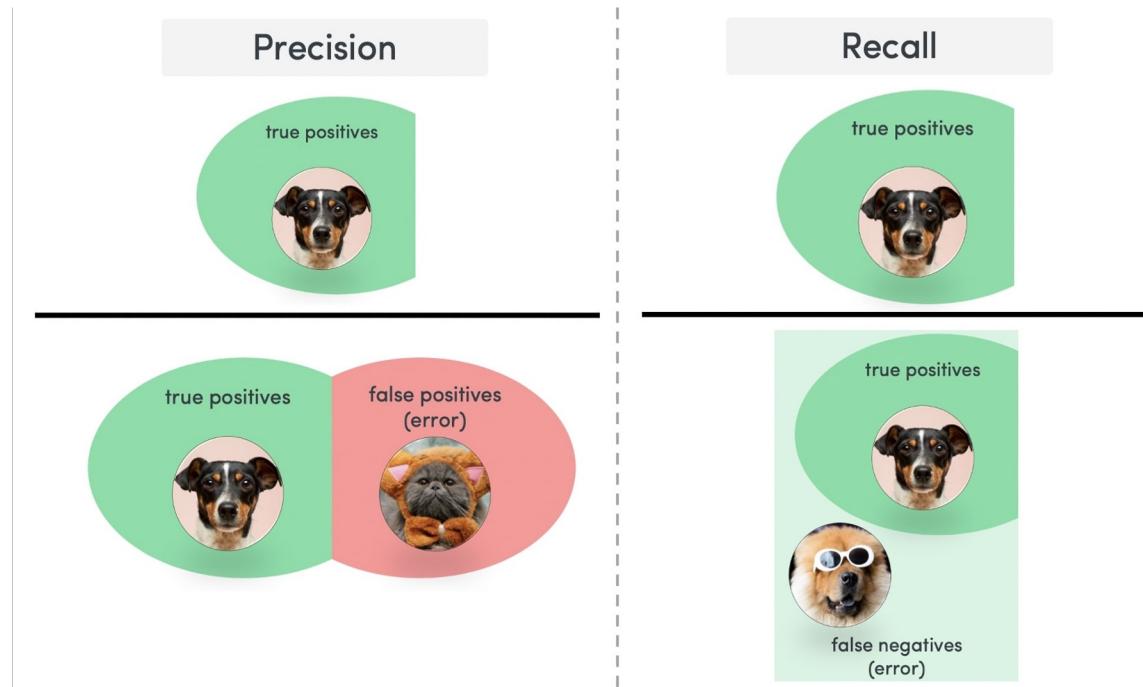
Selection = empirical question with model-specific solutions

4. Potentials and Limitations of LLMs in DH

Text Classification and Analysis: Evaluation

Metrics:

- Accuracy
- Precision
- Recall
- F1-Score



4. Potentials and Limitations of LLMs in DH

Other Use Cases

- Preprocessing
 - OCR, NER, ...
- Data Extraction
- Generation of
 - Summaries
 - Textual interpretations for data
 - Code

4. Potentials and Limitations of LLMs in DH

Challenges in Prompt Design

```
prompt = f"""
You are a literary scholar. _____ Occupation
Your task is to classify text excerpts on the basis of labels given to you.
```

Select one of the following labels to classify a text excerpt:

Label: process
Label: stative event
Label: non event

Let's think step by step. _____ Steps
Make sure to reach the right conclusion. _____ Conclusion
Use the following output format: _____ Structured Output
Label: str

Do NOT generate any more text or repeat the input!
Doing this task well is very important for my career. _____ Stakes
What event is the following Text snippet: {snippet}
Label:
"""

4. Potentials and Limitations of LLMs in DH

Challenges in Prompt Design

Model	Split	Best Acc.	conclusion	occupation	stakes	steps	structured output	template
CLAUDE	Test	0.64	True	False	True	True	True	Template4
GPT	Test	0.58	False	True	True	True	True	Template4
LLAMA-3	Test	0.55	False	True	True	True	True	Template3
MIXTRAL	Test	0.55	True	True	True	True	True	Template4
SAUERKRAUT	Test	0.54	True	True	True	True	True	Template4
LLAMA-2	Test	0.39	False	False	False	False	True	Template2
LEOLM	Test	0.33	True	True	True	False	True	Template3
CLAUDE	Eval	0.78	True	False	True	True	True	Template4
GPT	Eval	0.69	False	True	True	True	True	Template3
SAUERKRAUT	Eval	0.57	True	True	True	True	True	Template2
MIXTRAL	Eval	0.56	True	True	True	True	True	Template2
LLAMA-3	Eval	0.54	False	True	True	True	True	Template1
LLAMA-2	Eval	0.41	True	True	True	True	True	Template4
LEOLM	Eval	0.37	True	True	False	True	True	Template1

4. Potentials and Limitations of LLMs in DH

Challenges in Prompt Design

Prompt Templates might work for one model, but not for all!

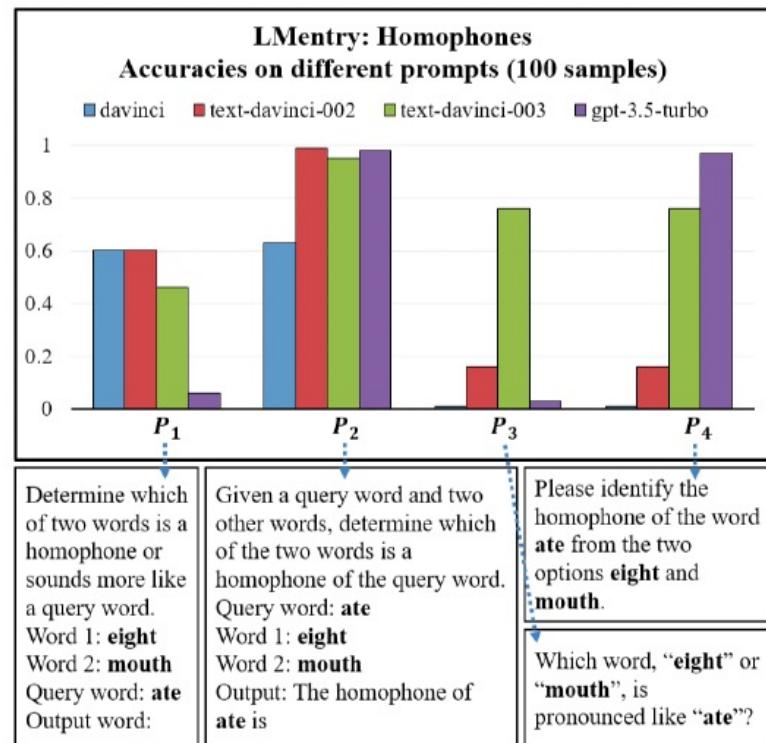
4. Potentials and Limitations of LLMs in DH

Challenges in Prompt Design

Prompt Brittleness

Mizrahi et al. 2024

- Comparing 20 LLMs over 39 tasks
- »Different instructions lead to statistically significant differences in performance for 21 out of the 25 tasks«



4. Potentials and Limitations of LLMs in DH

*Prompting LLMs does not circumvent the need
for
manually created reference data!*

4. Potentials and Limitations of LLMs in DH

Hallucinations and Bullshitting

Ethics and Information Technology (2024) 26:38

<https://doi.org/10.1007/s10676-024-09775-5>

ORIGINAL PAPER



ChatGPT is bullshit

Michael Townsend Hicks¹ · James Humphries¹ · Joe Slater¹

Published online: 8 June 2024

© The Author(s) 2024

Abstract

Recently, there has been considerable interest in large language models: machine learning systems which produce human-like text and dialogue. Applications of these systems have been plagued by persistent inaccuracies in their output; these are often called “AI hallucinations”. We argue that these falsehoods, and the overall activity of large language models, is better understood as *bullshit* in the sense explored by Frankfurt (On Bullshit, Princeton, 2005): the models are in an important way indifferent to the truth of their outputs. We distinguish two ways in which the models can be said to be bullshitters, and argue that they clearly meet at least one of these definitions. We further argue that describing AI misrepresentations as bullshit is both a more useful and more accurate way of predicting and discussing the behaviour of these systems.

Keywords Artificial intelligence · Large language models · LLMs · ChatGPT · Bullshit · Frankfurt · Assertion · Content

4. Further Considerations

Hallucinations and Bullshitting

Types of Bullshit (Hicks et al. 2024):

- *Bullshit (general)*: Any utterance produced where a speaker has indifference towards the truth of the utterance.
- *Hard bullshit*: Bullshit produced with the intention to mislead the audience about the utterer's agenda.
- *Soft bullshit*: Bullshit produced without the intention to mislead the hearer regarding the utterer's agenda.

4. Further Considerations

›open‹ vs. ›closed‹ source models

- Standards for scientific experiments:
 - Reproducibility of experiments
 - Knowledge of influencing factors
- Most LLMs are not fully documented
 - Liesenfeld et al. 2023
 - »while there is a fast-growing list of projects billing themselves as ›open source‹, many inherit undocumented data of dubious legality, few share the all-important instruction-tuning [...], and careful scientific documentation is exceedingly rare«
 - 13 features: Open code, LLM data, LLM weights, RLHF data, RLHF weights, License, Code, Architecture, Preprint, Paper, Data sheet, Package, API

4. Further Considerations

›open‹ vs. ›closed‹ source models

<https://opening-up-chatgpt.github.io>



Opening up ChatGPT: tracking openness of instruction-tuned LLMs

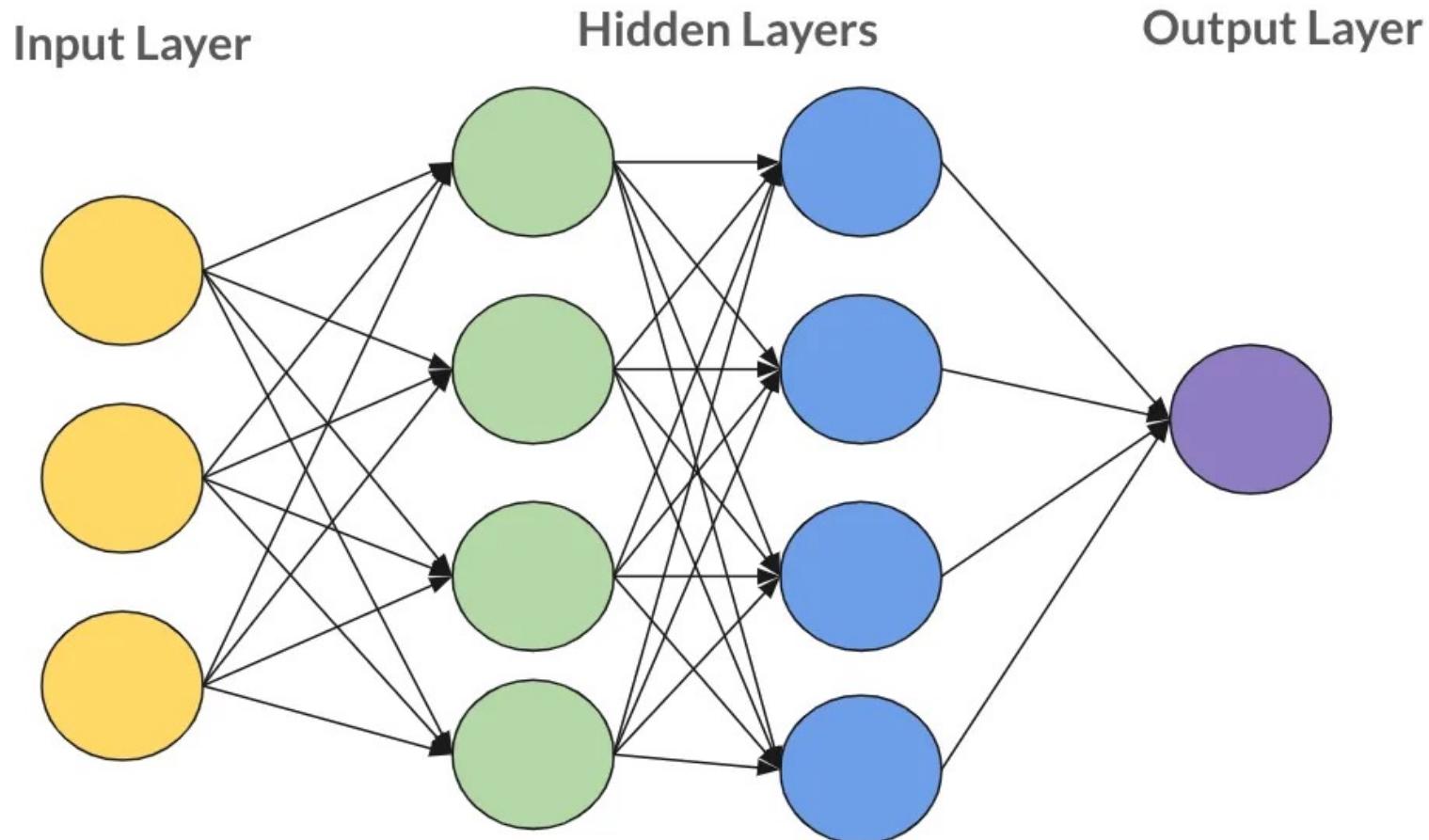
⚡ FAccT'24 paper ⚡ Liesenfeld, Andreas, and Mark Dingemanse. 2024. 'Rethinking Open Source Generative AI: Open-Washing and the EU AI Act'. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency (FAccT '24)*. Rio de Janeiro, Brazil: ACM. ([PDF](#)).

There is a growing amount of instruction-tuned text generators billing themselves as 'open source'. How open are they really? 🔗 FAccT'24 🔗 CUI'23 🔗 repo

Project (maker, bases, URL)	Availability						Documentation					Access			
	Open code	LLM data	LLM weights	RL data	RL weights	License	Code	Architecture	Preprint	Paper	Modelcard	Datasheet	Package	API	
OLMo 7B Instruct AllenAI	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	~	
	LLM base: OLMo 7B						RL base: OpenInstruct						12.5		
BLOOMZ bigscience-workshop	✓	✓	✓	✓	~	~	✓	✓	✓	✓	✓	✓	✗	✓	
	LLM base: BLOOMZ, mT0						RL base: xP3						12.0		
AmberChat LLM360	✓	✓	✓	✓	✓	✓	~	~	✓	✗	~	~	✗	✓	
	LLM base: Amber						RL base: ShareGPT + Evol-Instruct (sy...)						10.0		
Open Assistant LAION-AI	✓	✓	✓	✓	✗	✓	✓	✓	✓	~	✗	✗	✗	✓	
	LLM base: Pythia 12B						RL base: OpenAssistant Conversations						9.5		
OpenChat 3.5 7B Tsinghua University	✓	✗	✓	✗	✓	✓	~	✓	✓	✓	~	✗	✓	~	
	LLM base: Mistral 7B						RL base: ShareGPT with C-RLFT						9.5		
Pythia-Chat-Base-7... togethercomputer	✓	✓	✓	✓	✗	✓	✓	✓	~	✗	~	~	✓	✗	
	LLM base: EleutherAI pythia						RL base: OIG						9.5		
Cerebras GPT 111... Cerebras + Schramm	~	✓	✓	✓	✓	✓	~	✗	✓	~	✗	✓	✗	✓	
	LLM base: Cerebras						RL base: Alpaca (synthetic)						8.5		
RedPajama-INCITE... TogetherComputer	~	✓	✓	✓	✓	✓	~	~	~	✗	✗	✓	✓	✗	
	LLM base: RedPajama-INCITE-7B-Base						RL base: various (GPT-JT recipe)						8.5		
dolly databricks	✓	✓	✓	✓	✗	✓	✓	✓	✓	~	✗	✗	✗	✓	
	LLM base: EleutherAI pythia						RL base: databricks-dolly-15k						8.5		

4. Further Considerations

Size and Cost: Parameters of a Neural Network



Source: https://medium.com/@icaro_vazquez/neural-network-hyperparameter-optimization-with-hyperopt-f3e0cb4346c8

4. Further Considerations

Size and Cost

Date	Model	Parameters	Training Tokens
06/2018	GPT	117M	-
10/2018	BERT (large)	340M	3.3B
02/2019	GPT-2	1.5B	10B
10/2019	T5	11B	34B
05/2020	GPT-3	175B	300B
04/2022	PaLM	540B	768B
07/2022	BLOOM	175B	350B
02/2023	LlaMA	65B	1.4T
03/2023	GPT-4	-	-
04/2024	Llama 3	8B/70B	15T

4. Further Considerations

Size and Cost

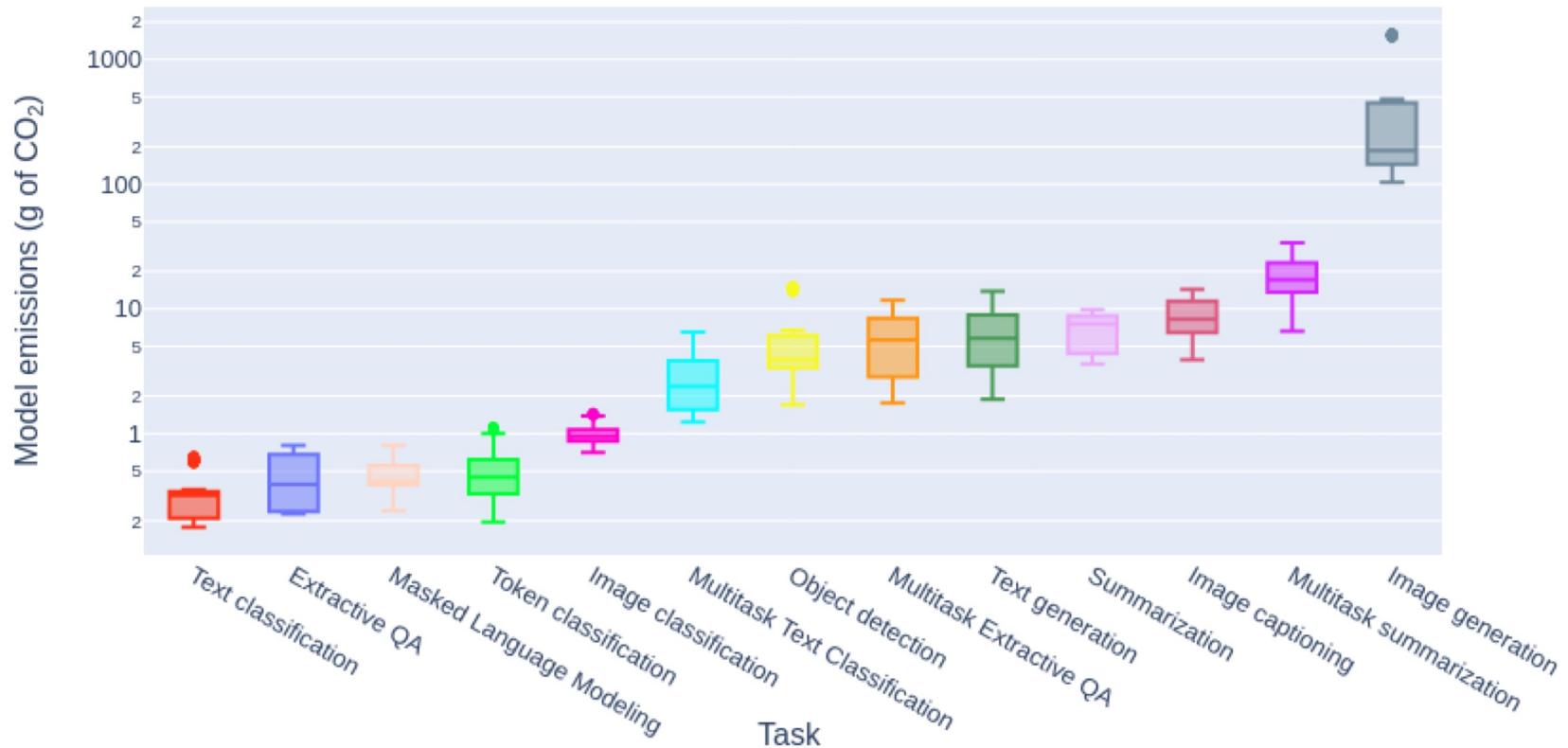
Craig Smith: »What LLMs Cost – There is no Free AI Lunch«

(<https://www.forbes.com/sites/craigsmith/2023/09/08/what-large-models-cost-you--there-is-no-free-ai-lunch/>)

- »When asked at an MIT event in July whether the cost of training foundation models was on the order of \$50 million to \$100 million, OpenAI's cofounder Sam Altman answered that it was ›more than that‹ and is getting more expensive.«
- »Estimates suggest that in January 2023, ChatGPT used nearly 30,000 GPUs to handle hundreds of millions of daily user requests. Sajjad Moazeni, a University of Washington assistant professor of electrical and computer engineering, says those queries may consume around 1 GWh each day, the equivalent of the daily energy consumption for about 33,000 U.S. households.«

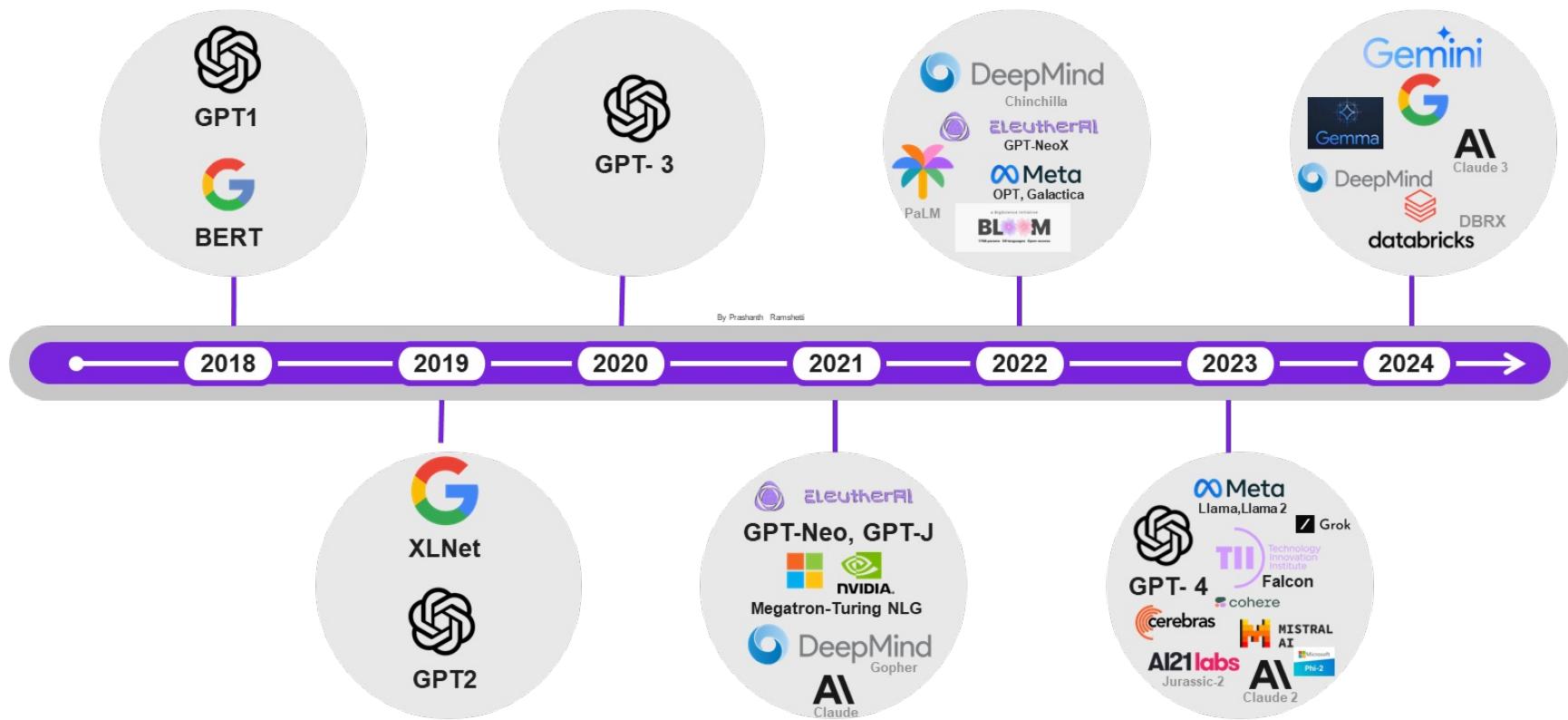
4. Further Considerations

Ecological impact: Luccioni et al. 2023



Source: <https://arxiv.org/abs/2311.16863>

5. Getting Started with LLMs: The LLM Landscape



5. Getting Started with LLMs

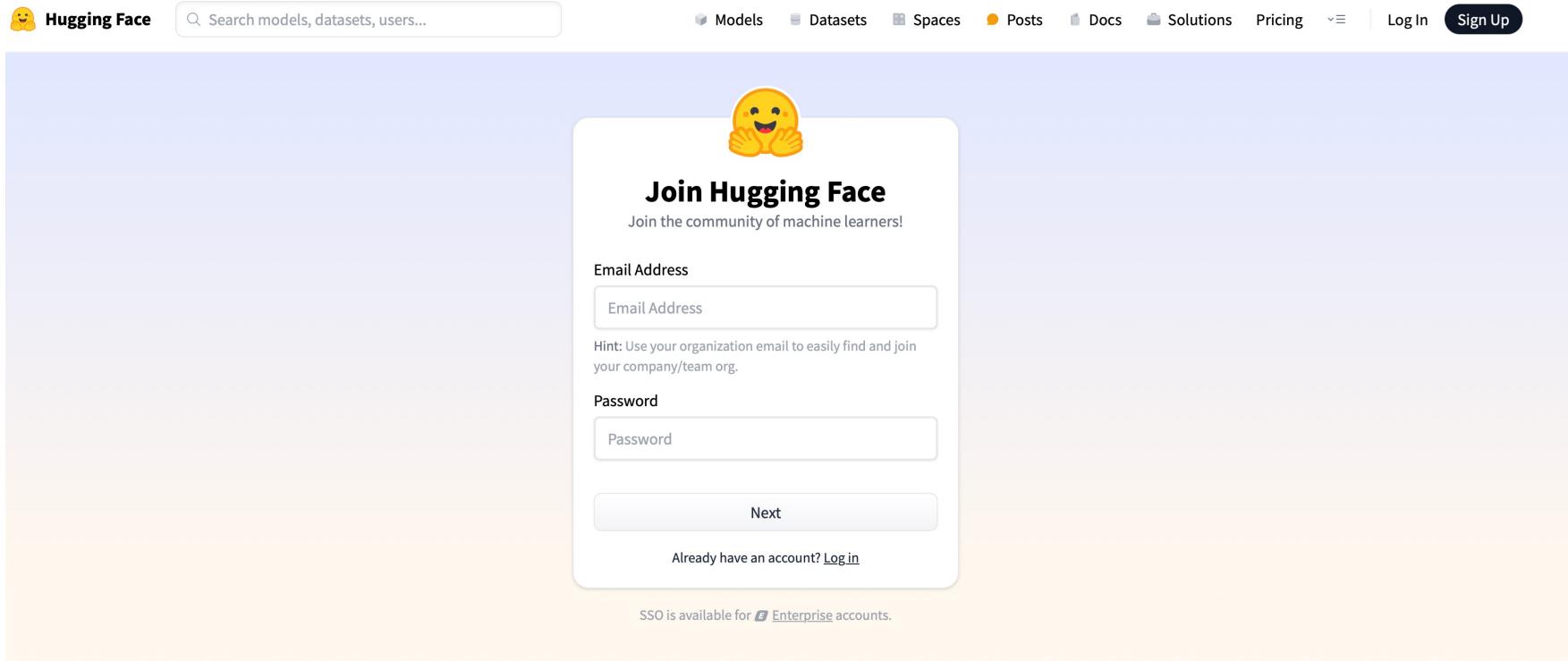
5. Getting Started with LLMs

[http://https://huggingface.co](https://huggingface.co)

The screenshot shows the Hugging Face website homepage. At the top, there is a navigation bar with a logo, a search bar, and links for Models, Datasets, Spaces, Posts, Docs, Pricing, Log In, and Sign Up. A banner at the top left says "NEW! AI Tools are now available in HuggingChat". The main content area features a large yellow emoji of hands clasped together, followed by the text "The AI community building the future." Below this, a subtext reads "The platform where the machine learning community collaborates on models, datasets, and applications." To the right, there is a sidebar titled "Tasks" with categories like Multimodal, Computer Vision, Natural Language Processing, Audio, Tabular, and Reinforcement Learning. The "Models" section on the right lists various pre-trained models such as "meta-llama/Llama-2-7b", "stabilityai/stable-diffusion", "openchat/openchat", "llyasviel/ControlNet-v1.1", "cerspense/zeroscope_v2_X", "meta-llama/Llama-2-13b", "tiiuae/falcon-40b-instruct", "WizardLM/WizardCoder-15B-V1.1", "CompVis/stable-diffusion-v1-4", "stabilityai/stable-diffusion-2", and "Salesforce/xgen-7b-8k-inst". Each model entry includes a thumbnail, name, description, and update information.

5. Getting Started with LLMs

Singing Up on HuggingFace



5. Getting Started with LLMs

Create an Access Token for Colab

The screenshot shows the Hugging Face Hub user interface. On the left, there's a sidebar with a profile picture of Axel Pichler (apichler) and a list of options: Profile, Account, Authentication, Organizations, Billing, and Access Tokens (which is currently selected). Below these are SSH and GPG Keys, Webhooks, Papers, Notifications, Local Apps and Hardware (marked as NEW), Gated Repositories, Content Preferences, Connected Apps, and Theme.

The main content area is titled "Access Tokens" and contains a section for "User Access Tokens". It explains that access tokens authenticate your identity to the Hugging Face Hub and allow applications to perform specific actions. It advises users to never post their tokens publicly. Two tokens are listed: "DataBricks_Course" (with a READ scope) and "Colab" (with a READ scope). Both tokens have a "Manage" button and a note to refresh the token to copy its value. A "New token" button is also present.

At the bottom of the page, there's a link to "Upgrade to Pro".

5. Getting Started with LLMs

Set Colab Secret

The screenshot shows the Google Colab interface with the title "DiDip_2024_hands-on.ipynb". The "Secrets" section is open, displaying configuration instructions and a table of existing secrets.

Instructions:

- Icon: 🔎 Konfigurieren Sie Ihren Code, indem Sie Umgebungsvariablen, Dateipfade oder Schlüssel speichern. Die hier gespeicherten Werte sind privat und nur für Sie und auf von Ihnen ausgewählten Notebooks sichtbar.
- Icon: 🗂️ Der geheime Name darf keine Leerzeichen enthalten.

Table of Existing Secrets:

Notebook-Zugriff	Name	Wert	Aktionen
📅	GEM_TOKEN	👁️ 📁 🗑️
📅	HF_TOKEN	👁️ 📁 🗑️
📅	ANTHROPIC_AP	👁️ 📁 🗑️

Add New Secret: + Neues Secret hinzufügen

Zugriff auf Ihre geheimen Schlüssel in Python über:

```
from google.colab import userdata  
userdata.get('secretName')
```