From Text to Threads

Large Language Models and their impact on the HEP community

Andrea Valenzuela Ramírez
CMS Offline & Computing
15th April 2024

inverted CSC
CERN School of Computing

Objectives

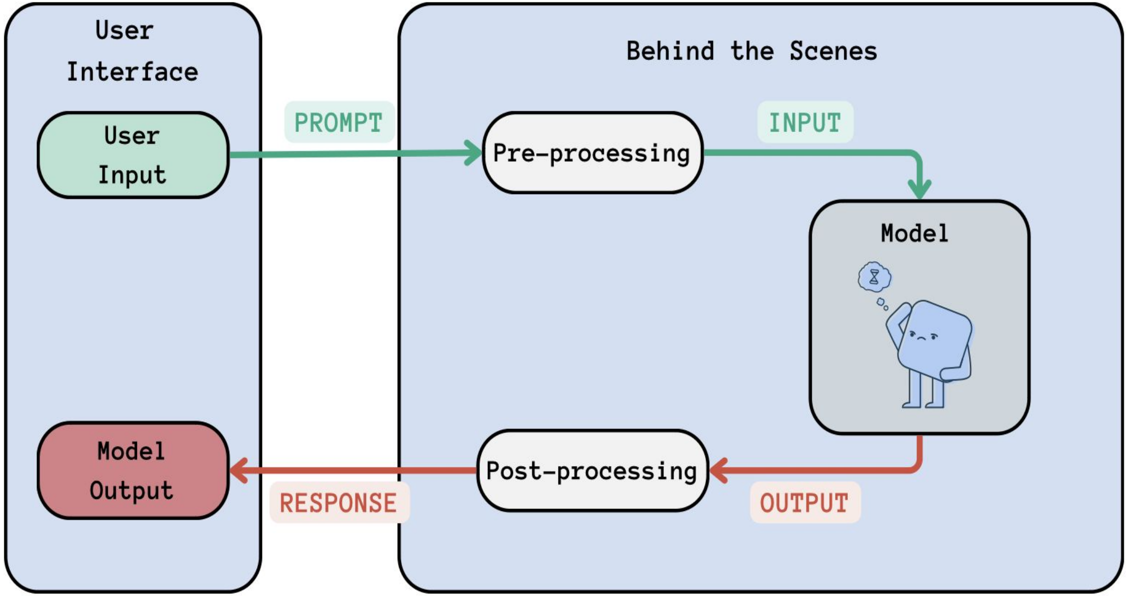
- Learning **how LLMs work** more in detail.
  - Key components.
  - Base model.
- **Prompt engineering strategies** to improve the LLM output.
- Emergence of **LLMs in the HEP** community.
- **LLMs for coding**.
  - Challenges.
  - Prompt engineering strategies.

GLOSSARY

LLM - Large Language Model
GPT - Family of LLMs from OpenAI (powering ChatGPT)

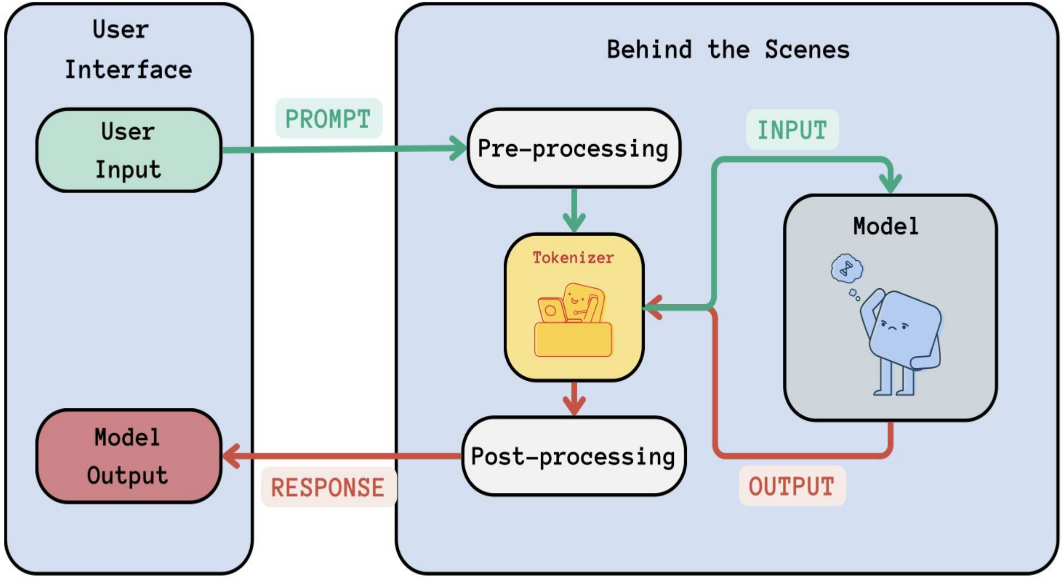
Introduction

What happens "behind the scenes" of a Large Language Model like ChatGPT?



Introduction

What happens "behind the scenes" of a Large Language Model like ChatGPT?

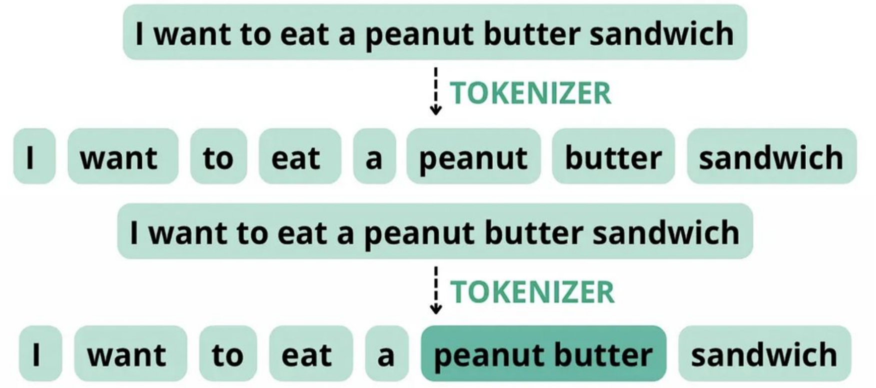


- LLMs work on **numerical data**.
- Tokenizer plays a crucial role since it has a direct impact on the model input.

LLM Tokenizer

There are different tokenization levels:

- Word-level
- Subword-level
- Character-level
- Byte Pair Encoding (BPE)



Different models use different tokenizers:

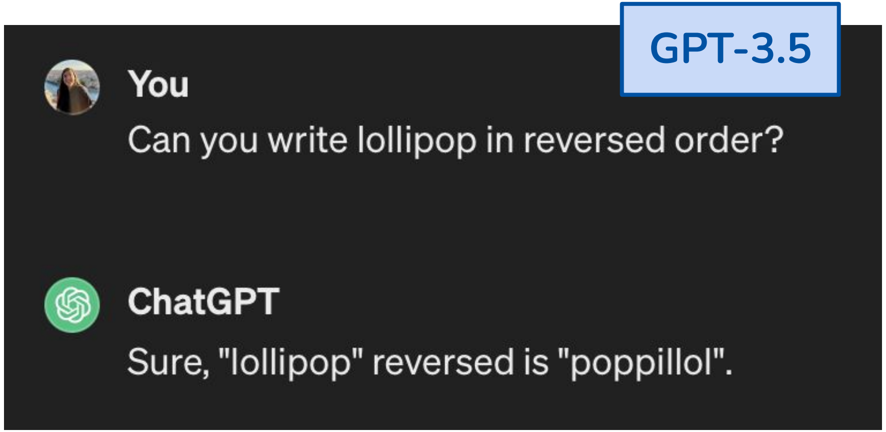
Encoding name	OpenAI models
`cl100k_base`	`gpt-4`, `gpt-3.5-turbo`, `text-embedding-ada-002`
`p50k_base`	Codex models, `text-davinci-002`, `text-davinci-003`
`r50k_base` (or `gpt2`)	GPT-3 models like `davinci`

tiktoken 0.6.0

```
pip install tiktoken
```

LLM Tokenizer

- Some completion errors occur due to the tokenizer and not because of the model's capabilities.




```
gpt2: 3 tokens
token integers: [75, 692, 42800]
token bytes: [b'l', b'oll', b'ipop']

p50k_base: 3 tokens
token integers: [75, 692, 42800]
token bytes: [b'l', b'oll', b'ipop']

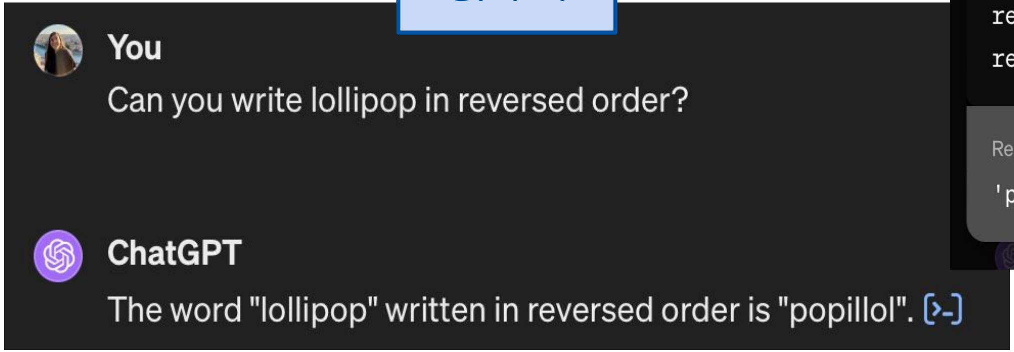
cl100k_base: 2 tokens
token integers: [75, 90644]
token bytes: [b'l', b'ollipop']
```

- The tokenizer mechanism is also the reason why LLMs are, in general, not good at following character restrictions. **LLMs count in tokens!**

LLM Tokenizer

- New iterations of GPT models have workarounds to overcome these limitations.

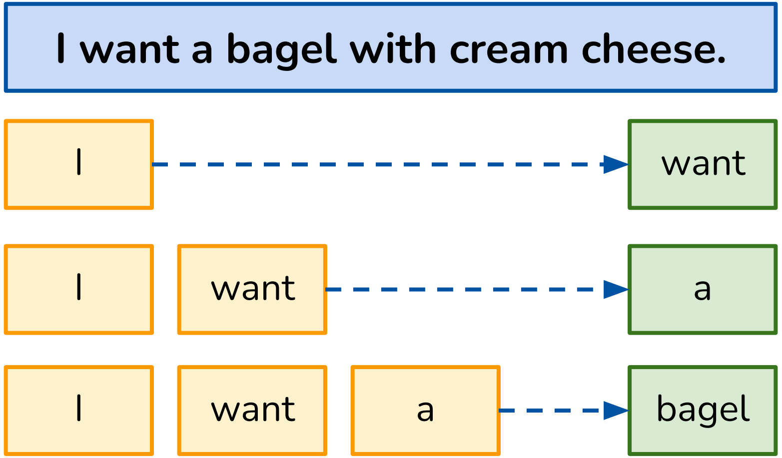
GPT-4



- Finally, tokenization also involves setting the rules on how tokens are mapped to numerical IDs based on the model's vocabulary, and then to **vector embeddings**.

Transformer Models

- LLMs are built using a specific part of the **Transformer Architecture.**
  - Decoder-only model.
- Training begins with **Self-supervised Learning**.
  - Training sets are constructed by breaking down a sentence into a series of training examples.
- The core of this architecture is the **Attention Mechanism**.
  - Weight the “importance” of the different input words.



See Attention Mechanism in action here!

Instruction-tuned LLMs

- **Base LLMs** are trained to predict the next token based on training data.

Once upon a time, there was a unicorn **that lived in a magical forest with all her friends**	What is the capital of France? **What is France's largest city?**

- **Instruction-tuned LLMs** are fine-tuned on data comprised by examples of where the output follows an input instruction.

> What is the capital of France?
> **Paris**

  - **[EXTRA STEP]:** Reinforcement Learning from Human Feedback
    Obtain human ratings of the quality of the LLMs outputs and tune the model to increase the probability of generating higher rated outputs.

Prompt Engineering

There are strategies that users can adopt to improve the LLMs output.

- **Positive / Negative prompting**
  - Instruct the model on what to focus on.
  - “Do X, don’t do Y”.
- **Ask for a structured output**
  - Directing the model on the desired output format.
  - Standardizing the model output.

```
prompt = f"""
    Given a collection of e-commerce reviews your task is to determine
    the sentiment of each review.

    The reviews are given in a numbered list delimited by 3 backticks,

    ```{reviews}```

    Output True if the review is Positive and False if it is Negative.
"""

response = chatgpt_call(prompt)
print(response)
```

```
True
False
True
True
False
```

Prompting as an iterative approach
```

Prompt Engineering

```python
prompt = f"""
    Given a collection of e-commerce reviews your task is to determine
    the sentiment of each review.

    The reviews are given in a numbered list delimited by 3 backticks, i.e. ```.

    ```{reviews}```

    Output only if the review is Negative or Positive and a brief summary of the review.
    Use only one sentence for the summary.

    Give your response in a HTML table with the review number, the sentiment and the summary.

"""

response = chatgpt_call(prompt)

from IPython.display import display, HTML
display(HTML(response))
```

Review Number	Sentiment	Summary
1	Positive	The children's computer is fantastic and highly recommended.
2	Negative	The children's computer didn't live up to expectations and the delivery was delayed.
3	Positive	The children's computer is a great educational toy, but the delivery took longer than expected.
4	Positive	The children's computer is highly interactive and keeps kids entertained for hours.
5	Negative	The children's computer arrived damaged and some features didn't work properly, with delayed delivery.

Models can handle concrete structured output in formats such as **Markdown**, **HTML** or **JSON**.
```

Prompt Engineering

❖ **Specify intermediate steps**

- Provide a chain of relevant reasonings to follow for reaching the answer.
- Compute intermediate steps implies spending **more computational effort.**
- Sometimes you don't even need to define the intermediate steps: [*"Let's think step by step"*]

Human (without principle):
What is the derivative of x^5 + x^9 - x^3 +1?

LLaMA-2-13B: To find the derivative of x^5 + x^9 - x^3 + 1, you can use the power rule of differentiation, which states that the derivative of x^n is equal to n * x^(n-1). Starting with x^5 + x^9 - x^3 + 1, you can apply the power rule to the first two terms, which gives you: 5 * x^(5-1) + 9 * x^(9-1) You can then simplify the expression by combining like terms: 5 * x^4 + 9 * x^8 Finally, you can simplify the expression further by canceling out the common factor of x^4: 5 * 9 * x^4 = 45 * x^4

Therefore, the derivative of x^5 + x^9 - x^3 + 1 is equal to 45 * x^4.

Human (with principle):
What is the derivative of x^5 + x^9 - x^3+1? let's think step by step

LLaMA-2-13B: find the derivative of the function, we need to take the derivative of each term in the function. The derivative of x^5 is 5x^4. The derivative of x^9 is 9x^8. The derivative of x^3 is 3x^2. The derivative of 1 is 1. Therefore, the derivative of the function is: 5x^4 + 9x^8 - 3x^2 + 1 = 5x^4 + 9x^8 - 3x^2 + 1 - (1) = 5x^4 + 9x^8 - 3x^2 Therefore, the derivative of the function x^5 + x^9 - x^3+1 is 5x^4 + 9x^8 - 3x^2.

[source] Bsharat, S. M., Myrzakhan, A., & Shen, Z. (2023). **Principled Instructions Are All You Need for Questioning LLaMA-1/2, GPT-3.5/4**. *arXiv preprint arXiv:2312.16171*.

Prompt Engineering

❖ Few-shot prompting

- Jason Wei and Denny Zhou et al. (Google) studied it in 2022.
- It can be seen as **demonstrating the model how to answer with a few examples**.
- It can also help correcting some model errors or biases.

More examples here!

```python
import openai

def chatgpt_call(prompt, model="gpt-3.5-turbo"):
    response = openai.ChatCompletion.create(
        model=model,
        messages=[{"role": "user", "content": prompt}]
    )
    return response.choices[0].message["content"]
```

```python
chatgpt_call("Teach me about optimism. Keep it short.")
```

```
'Optimism is a mental attitude that focuses on the positive a
spects of a situation and believes that good things will happ
en in the future. It involves having a positive outlook, bein
g hopeful, and looking for solutions rather than dwelling on
problems. Optimistic people tend to be happier, healthier, an
d more successful in achieving their goals. Practicing gratit
ude, positive thinking, and surrounding yourself with support
ive people can help cultivate optimism in your life.'
```

Prompt Engineering

❖ Few-shot prompting

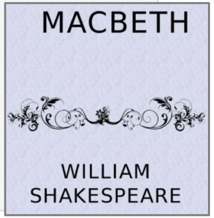
- Jason Wei and Denny Zhou et al. (Google) studied it in 2022.
- It can be seen as **demonstrating the model how to answer with a few examples**.
- It can also help correcting some model errors or biases.

```
prompt = """
Your task is to answer in a consistent style.

<user>: Teach me about ambition.

<system>: I have no spur\n
To prick the sides of my intent, but only\n
Vaulting ambition, which o'erleaps itself\n
And falls on the other.\n

<user>: Teach me about optimism.
"""
```




```
chatgpt_call(prompt)
```

```
"<system>: Optimism is like a bright beacon of hope\n\nGuidin
g you through life's darkest days\n\nIt's a mindset that sees
opportunities\n\nAnd believes in better ways."
```

More examples here!

Prompt Engineering

❖ **Few-shot prompting**

```
sql_tables = """
CREATE TABLE PRODUCTS (
    product_name VARCHAR(100),
    price DECIMAL(10, 2),
    discount DECIMAL(5, 2),
    product_type VARCHAR(50),
    rating DECIMAL(3, 1),
    product_id VARCHAR(100)
);

CREATE TABLE ORDERS (
    order_number INT PRIMARY KEY,
    order_creation DATE,
    order_status VARCHAR(50),
    product_id VARCHAR(100)
);
```

```
prompt = f"""
Given the following SQL tables, your job is to provide the required SQL tables
to fulfill any user request.

Tables: <{sql_tables}>

User request: ```{user_input}```
"""
```

```
user_input = """
What model of TV has been sold the most in the store?
"""
```

```
response = chatgpt_call(prompt)
print(response)
```

```
To fulfill the user request, we need to find the model of TV that has been sold
the most in the store. We can achieve this by joining the PRODUCTS and ORDERS t
ables and grouping the results by the product model. Here is the required SQL q
uery:

```sql
SELECT p.product_name AS model_of_tv, COUNT(*) AS total_sold
FROM PRODUCTS p
JOIN ORDERS o ON p.product_id = o.product_id
WHERE p.product_type = 'TVs' AND o.order_status = 'Delivered'
GROUP BY p.product_name
ORDER BY total_sold DESC
LIMIT 1;
```
```

Default queries have no format!

Prompt Engineering

❖ Few-shot prompting

- Let's provide 4 examples of how I like formatting my queries:

```
fewshot_examples = """
User: What is the latest order that has been issued?
System: You first need to join both orders and products
tables and filter by the latest order_creation datetime:

SELECT
      P.product_name AS model_of_tv
FROM products AS P
JOIN orders AS O
  ON P.product_id = O.product_id

WHERE O.order_creation = (SELECT MAX(order_creation) FROM orders)
GROUP BY p.product_name
LIMIT 1;
```

+3 more examples

```
user_input = """
What model of TV has been sold the most in the store?
"""
```

```
response = chatgpt_call(prompt)
print(response)
```

```
System: You first need to join both orders and products tables, filter only tho
d to TVs, and count the number of orders that have been issued:

SELECT
       P.product_name AS model_of_tv,
       COUNT(*)       AS total_sold
FROM products AS P
JOIN orders    AS O
  ON P.product_id = O.product_id

WHERE P.product_type = 'TVs'
GROUP BY P.product_name
ORDER BY total_sold DESC
LIMIT 1;
```

Prompt Engineering

❖ Few-shot prompting

```
user_input = """
What product is the one that takes longer to deliver?
"""
```

```
response = chatgpt_call(prompt)
print(response)
```

```
System: You need to join both orders and products tables, calculate the delivery time for each order, a
nd select the product with the maximum delivery time:

SELECT
    P.product_name AS product_with_longest_delivery
FROM
    products AS P
JOIN
    orders AS O ON P.product_id = O.product_id
WHERE
    O.order_status = 'Delivered'
ORDER BY
    O.delivery_date - O.order_creation DESC
LIMIT 1;
```

Prompt Engineering

❖ Few-shot prompting

```
user_input = """
What product is the one that takes longer to deliver?
"""
```

```
response = chatgpt_call(prompt)
print(response)
```

```
System: You need to join both orders and products tables, calculate the delivery time for each order, a
nd select the product with the maximum delivery time:

SELECT
    P.product_name AS product_with_longest_delivery
FROM
    products AS P
JOIN
    orders AS O ON P.product_id = O.product_id
WHERE
    O.order_status = 'Delivered'
ORDER BY
    O.delivery_date - O.order_creation DESC
LIMIT 1;
```

It directly computes the difference between two datetime SQL variables, which for most SQL versions and platforms does not work (*SQLite*).

In this case, I like using the **julianday()** function.

Prompt Engineering

❖ Few-shot prompting

- It is also possible to correct model errors by providing examples.

```
fewshot_examples += """
User: Compute the time that it takes to delivery every product?
System: You first need to join both orders and products tables,
filter only those orders that have been delivered and compute
the difference between both order_creation and delivery_date.:

SELECT
    P.product_name AS product_with_longest_delivery,
    julianday(O.delivery_date) - julianday(O.order_creation) AS TIME_DIFF

FROM
    products AS P
JOIN
    orders AS O ON P.product_id = O.product_id
WHERE
    O.order_status = 'Delivered';


"""
```

```
user_input = """
What product is the one that takes longer to deliver?
"""
```

```
response = chatgpt_call(prompt)
print(response)
```

```
System: You first need to join both orders and products tables, filter only those orders that have been
delivered, compute the difference between the order_creation and delivery_date, and then select the pro
duct with the longest delivery time:

SELECT
    P.product_name AS product_with_longest_delivery,
    julianday(O.delivery_date) - julianday(O.order_creation) AS delivery_time

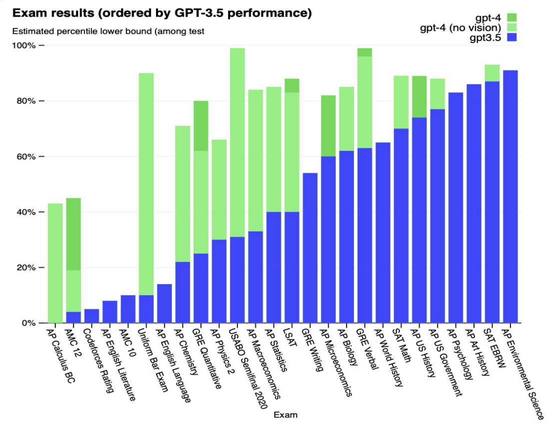
FROM
    products AS P
JOIN
    orders AS O ON P.product_id = O.product_id
WHERE
    O.order_status = 'Delivered'
ORDER BY delivery_time DESC
LIMIT 1;
```

LLMs in the HEP Community

❖ GPT models have made it to one of the key conferences for the HEP community: Conference on Computing in High Energy Physics and Nuclear Physics - **CHEP 2023.**

Evolution and Revolutions in Computing: Science at the Frontier - *David Dean (Jefferson Lab)*

❖ *Can ChatGPT do physics?*



❖ **Model Hallucinations**

LLMs sometimes produce outputs that are factually incorrect, unrealistic, or entirely fabricated, despite being presented in a confident manner.

ChatGPT will try to provide an answer to any of the given queries, even if it has not enough information about the target topic.

LLMs in the HEP Community

❖ Model Hallucinations

Problem solving as a translation task - François Charton (Meta AI)

Showing examples on Linear Algebra with Transformers.

“Are hallucinations predictable and principled or do models confabulate, fail at random?”

- Analyzing the distribution of error types, they found that **the model failed for good mathematical reasons**.
    - **It stays “roughly right”**: Some principles have been learnt. The task the model cannot perform is consistent.
    - **Failing in traditionally hard tasks**: ill-conditioned matrix **->** hard to invert.

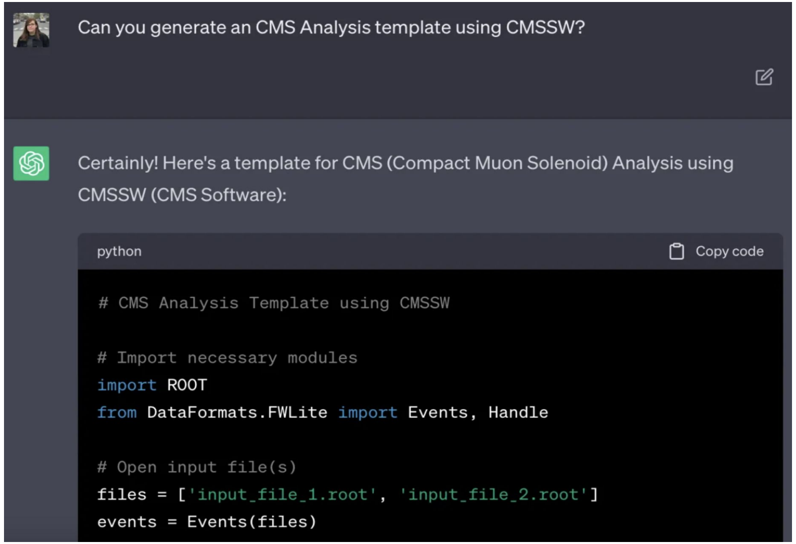
LLMs in the HEP Community

Radically different futures for HEP enabled by AI/ML - *Kyle Crammer (Wisconsin-Madison)*

- Proposing the introduction of ChatGPT as a valuable asset in the HEP toolkit. Concretely, as a **coding assistant**.

- Each experiment in the HEP community has its own coding templates that LLMs could learn to generate by fine-tuning strategies.

- Current GPT models already know about experiment-specific coding conventions.

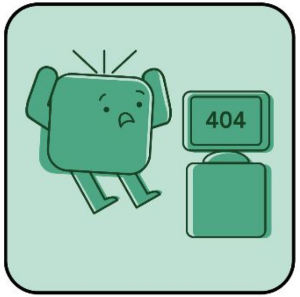


LLMs for Coding

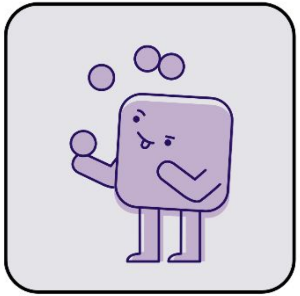
- The interest in using LLMs for coding has been rapidly raised and some have attempted to turn natural language generation into code generation.

- However, **LLMs are not good at coding** “out of the box” already showing some **issues** at an early stage:

Tokenizer



Context Windows



Training

LLMs for Coding - **Tokenizer**

The distribution of words in natural text is very different from that of coding.

- Strict syntax compared to natural language.
- Code often involves repetitive structures and patterns, such as loops and function calls, that are less common in natural language.

```
Input: "
def compare(str):
    """Prints a comparison."""
"

gpt2: 18 tokens
token bytes: [b'\n', b'def', b' compare', b'(', b'str', b'):', b'\n', b' ', b' ', b' ', b' """', b'Print',
b's', b' a', b' comparison', b'."', b'""', b'\n']

cl100k_base: 12 tokens
token bytes: [b'\n', b'def', b' compare', b'(str', b'):\n', b'    ', b' """', b'Print', b's', b' a', b' compa
rison', b'."""\n']
```

LLMs for Coding - **Tokenizer**

- One of the largest source of inefficiency arises from **encoding white-spaces**. Text tokenizers often treat indentation as mere whitespace:

```
Input: "
def compare(str):
    """Prints a comparison."""
"

gpt2: 18 tokens
token bytes: [b'\n', b'def', b' compare', b'(', b'str', b'):', b'\n', b' ', b' ', b' ', b' """', b'Print',
b's', b' a', b' comparison', b'."', b'""', b'\n']

p50k_base: 16 tokens
token bytes: [b'\n', b'def', b' compare', b'(', b'str', b'):', b'\n', b'   ', b' """', b'Print', b's', b'
a', b' comparison', b'."', b'""', b'\n']

cl100k_base: 12 tokens
token bytes: [b'\n', b'def', b' compare', b'(str', b'):\n', b'   ', b' """', b'Print', b's', b' a', b' compa
rison', b'."""\n']
```

(GPT text model)

(Codex)

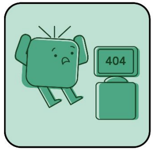
(GPT-4)

LLMs for Coding - **Context Windows**

- A context window refers to the amount of tokens the model can consider at any given time during its processing.

- Finite context windows make **challenging to generate consistent code** with the entire codebase.
  - Complex Code Dependencies
  - Long-Term Logical Structures

In natural language generation, finite context windows are normally managed by using **summarization**.



LLMs for Coding - **Training**

- General LLMs are trained for **left-to-right generation**, which implies predict the next token given a sequence of tokens

- Considering only the left context makes them **less powerful in coding tasks**.

COMMON CODING TASKS

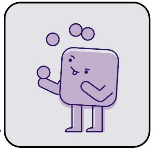
Code infilling (suggestions)

Renaming variables

Docstring generation

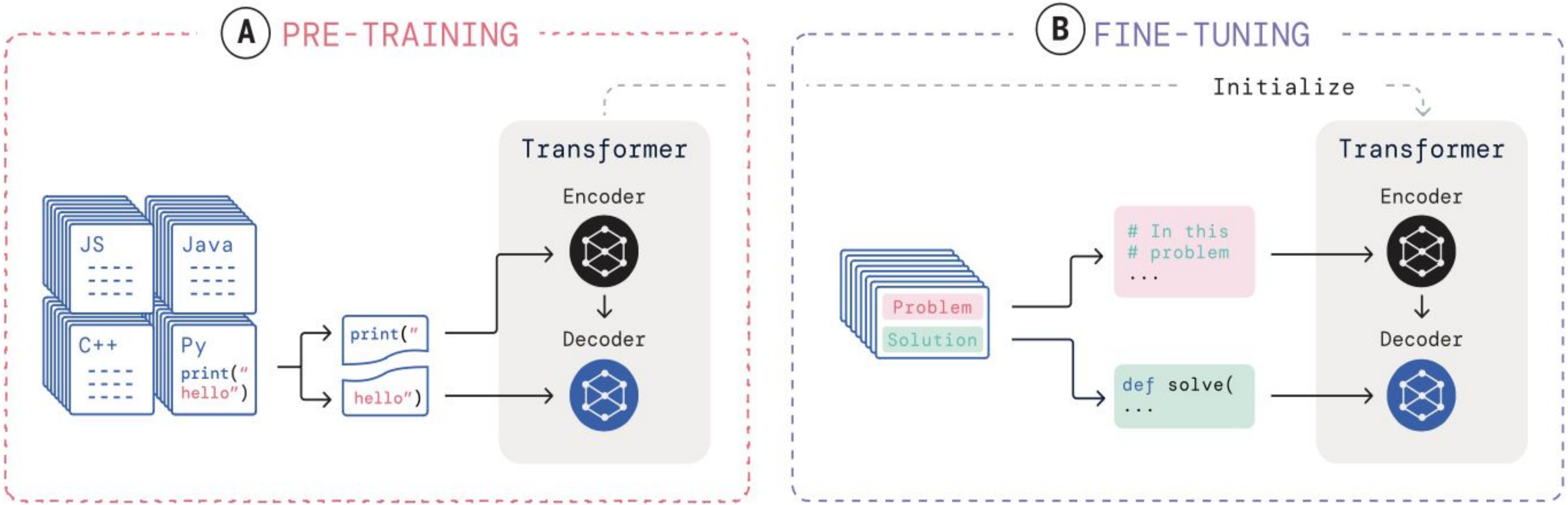
Return type prediction

```
def count_words(filename: str) -> Dict[str, int]:
    """Count the number of occurrences of each word in the file."""
    with open(filename, 'r') as f:
        word_counts = {}
        for line in f:
            for word in line.split():
                if word in word_counts:
                    word_counts[word] += 1
                else:
                    word_counts[word] = 1
    return word_counts
```



LLMs for Coding - **Training**

- Although left and right contexts are needed, most of the models rely on **left-context only and include fine-tuning in coding tasks** as part of the training.



[source] Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., ... & Vinyals, O. (2022). **Competition-level code generation with alphacode**. Science, 378(6624), 1092-1097.

LLMs for Coding - **Training**

❖ **InCoder** model proposes a Causal Masked Objective to incorporate right context during training [1].

```python
def count_words(filename: str) -> Dict[str, int]:
    """Count the number of occurrences of each word in the file."""
    with open(filename, 'r') as f:
        word_counts = {}
        for line in f:
            for word in line.split():
                if word in word_counts:
                    word_counts[word] += 1
                else:
                    word_counts[word] = 1
    return word_counts
```

Original Document

```python
def count_words(filename: str) -> Dict[str, int]:
    """Count the number of occurrences of each word in the file."""
    with open(filename, 'r') as f:
        <MASK:0> in word_counts:
                word_counts[word] += 1
            else:
                word_counts[word] = 1
    return word_counts
<MASK:0> word_counts = {}
        for line in f:
            for word in line.split():
                if word <EOM>
```

[source] [1]

Masked Document

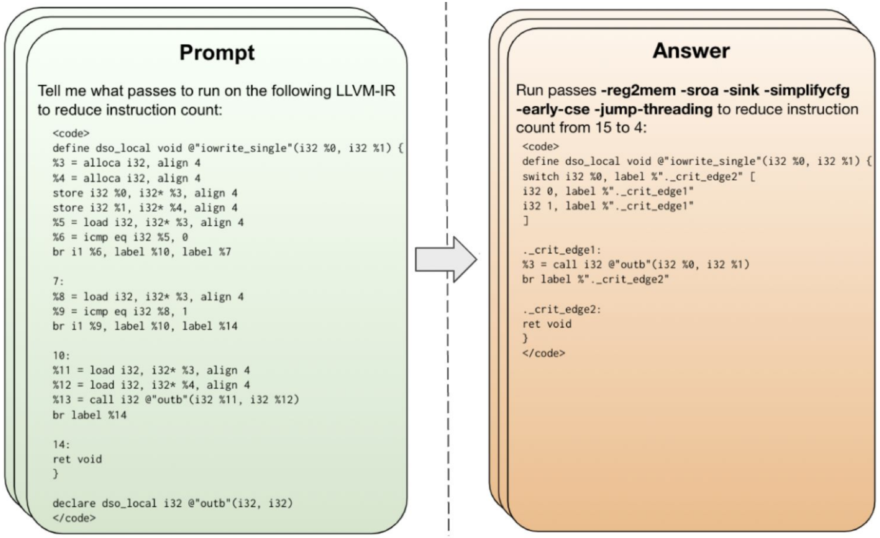
❖ **CodeCompose** [2] makes some modifications to the training objective of InCoder:

- Masking step to the language level instead to the tokenized text.
- Masking at trigger characters where the model will be queried during the inference.

LLMs for Coding - **Prompt Engineering**

- Explore the weak (and noisy) **pattern of natural and programming language** in code, e.g code comments [3][5].

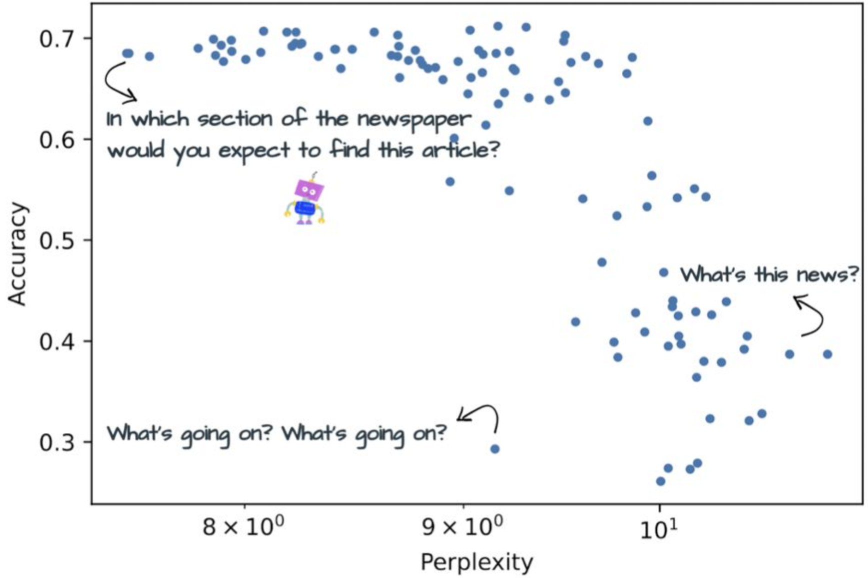
- Asking for **auxiliary learning tasks** improves the performance of the model [6].



[source] Cummins, C., Seeker, V., Grubisic, D., Elhoushi, M., Liang, Y., Roziere, B., ... & Leather, H. (2023). **Large language models for compiler optimization.** arXiv preprint arXiv:2309.07062.

LLMs for Coding - **Prompt Engineering**

- **Better prompt understanding** (with lower prompt perplexity as a proxy) leads to more functionally accurate programs [3].
- **Iterative decoding**, where the model can be used to refine its output [1] or **hierarchical models**.
- Taking **security** into account when executing AI–generated code [4].



[source] Gonen, H., Iyer, S., Blevins, T., Smith, N. A., & Zettlemoyer, L. (2022). Demystifying prompts in language models via perplexity estimation. *arXiv preprint arXiv:2212.04037*.

Final Remarks

- It is interesting to know how LLMs work before starting using them everywhere.

- I am not discouraging anyone to use LLMs.
  - Knowing their flaws can help when crafting our prompts to get the "best" completion for our use-case.
- LLMs for coding is an emerging topic with quite some research lines yet to be explored.

Final Remarks

- LLMs in science and in the HEP context are being exploited - for good or for bad.

Leveraging Language Models for Particle Reconstruction — *Xiangyang Ju*
Lecture Hall 2, Charles B. Wang Center, Stony Brook University — 16:50 - 17:10

Finetuning Foundation Models for Joint Analysis Optimization — *Matthias Vigl*
Lecture Hall 2, Charles B. Wang Center, Stony Brook University — 17:10 - 17:30

Beyond Language: Foundation Models for Collider Physics Data — *Anna Hallin*
Lecture Hall 2, Charles B. Wang Center, Stony Brook Un

Denoising Graph Super-Resolution with Diffusion Mo
Nilotpal Kakati

[ACAT]

PACuna: Automated Fine-Tuning of Language Models for Particle Accelerators (20'+10') — *Antonin Sulc*
Auditorium, DESY — 09:00 - 09:30

Building an Intelligent Accelerator Operations Assistant using Advanced Prompt Engineering Techniques and a High Level Control System Toolkit (20'+10')
Dr. Frank Mayet

Large Language Models for Particle Accelerator Tuning (20'+10') — *Jan Kaiser*
Auditorium, DESY — 10:00 - 10:30

AccGPT: A Vision for AI Assistance at CERN's Accelerator Control and Beyond (20'+10') — *Florian Rehm y otros*
Auditorium, DESY — 10:30 - 11:00

1st Large Language Models in Physics Symposium
LIPS
Hamburg, Feb 21 - 23, 2024

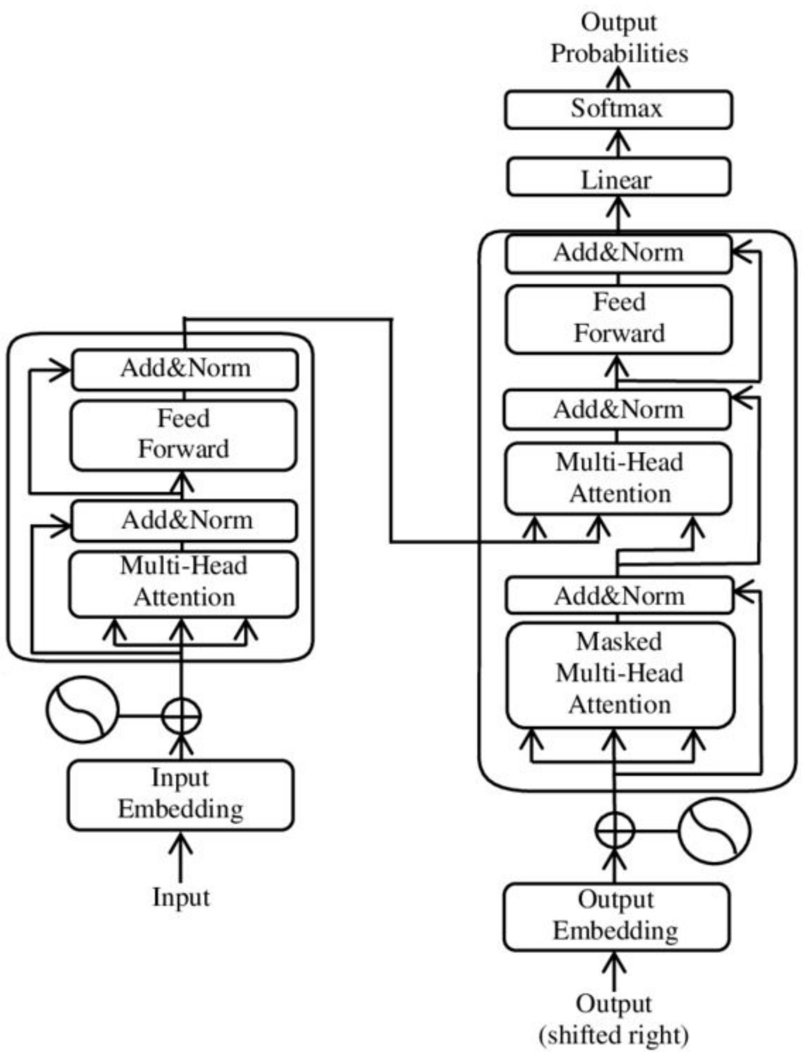
[LIPS]

Questions? :)

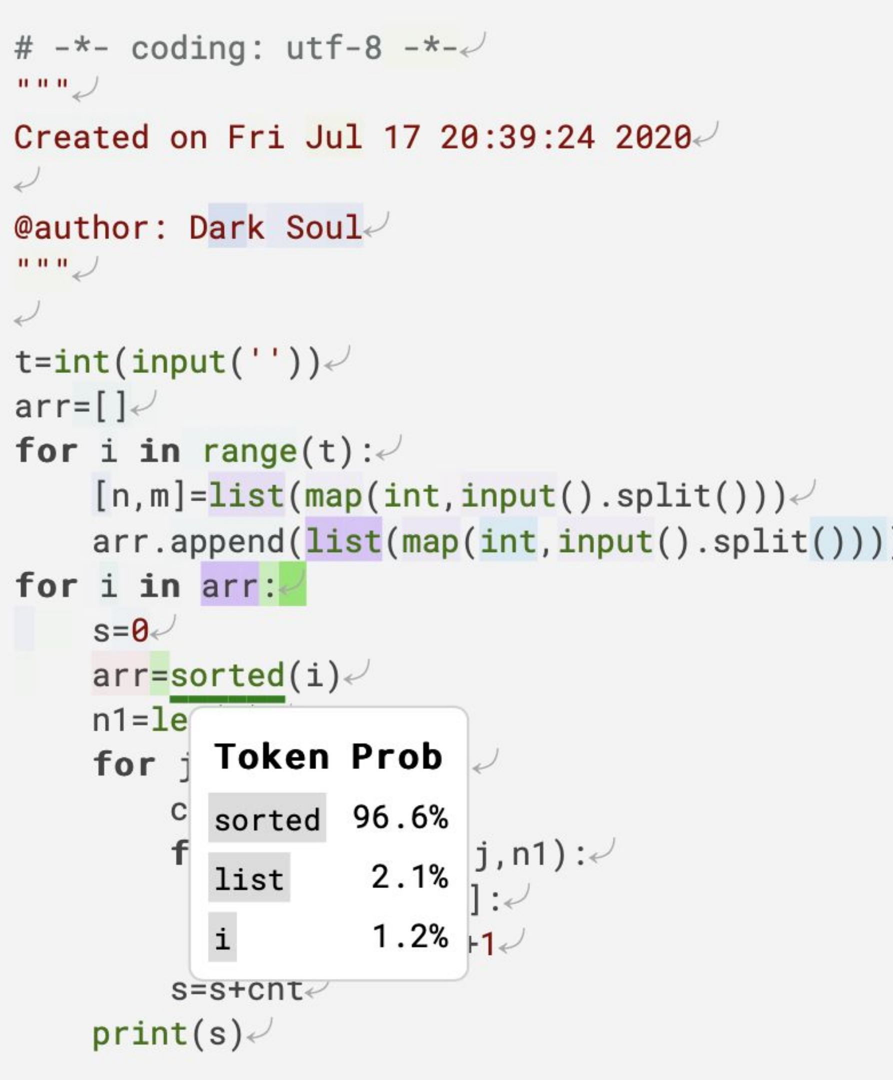
Andrea Valenzuela Ramírez

https://github.com/aandvalenzuela

andrea.valenzuela.ramirez@cern.ch



Back-up


-*- coding: utf-8 -*-
"""
Created on Fri Jul 17 20:39:24 2020
@author: Dark Soul
"""
t=int(input(''))
arr=[]
for i in range(t):
[n,m]=list(map(int,input().split()))
arr.append(list(map(int,input().split()))
for i in arr:
s=0
arr=sorted(i)
s=s+cnt
print(s)
Token Prob
sorted 96.6%
list 2.1%
i 1.2%

Coding Models

[1] Fried, D., Aghajanyan, A., Lin, J., Wang, S., Wallace, E., Shi, F., ... & Lewis, M. (2022). Incoder: A generative model for code infilling and synthesis. arXiv:2204.05999.

[2] Murali, V., Maddila, C., Ahmad, I., Bolin, M., Cheng, D., Ghorbani, N., ... & Nagappan, N. (2023). CodeCompose: A Large-Scale Industrial Deployment of AI-assisted Code Authoring. arXiv:2305.12050.

[3] Nijkamp, E., Pang, B., Hayashi, H., Tu, L., Wang, H., Zhou, Y., ... & Xiong, C. (2022). Codegen: An open large language model for code with multi-turn program synthesis. arXiv:2203.13474.

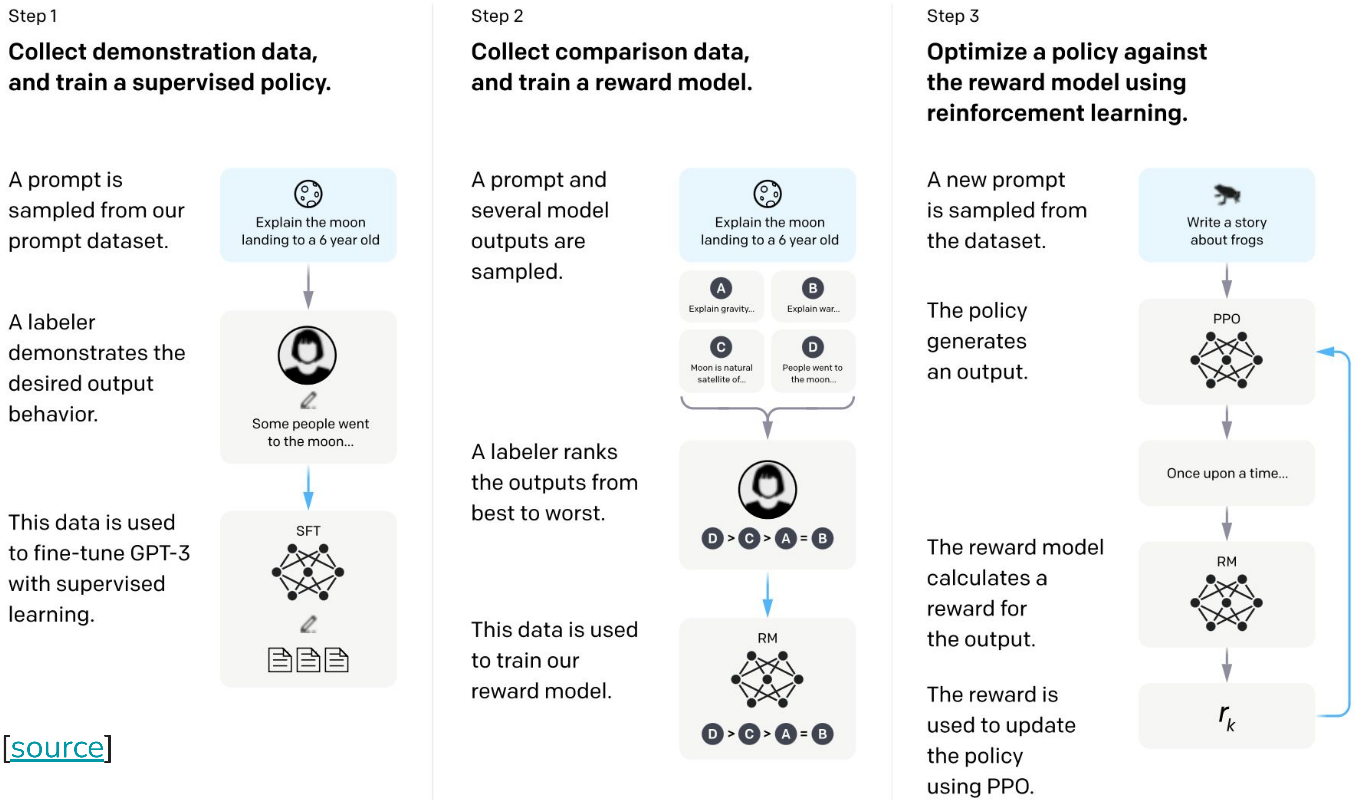
[4] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., ... & Zaremba, W. (2021). Evaluating large language models trained on code. arXiv:2107.03374.

[5] Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., ... & Vinyals, O. (2022). Competition-level code generation with alphacode. Science, 378(6624), 1092-1097.

—

[6] Cummins, C., Seeker, V., Grubisic, D., Elhoushi, M., Liang, Y., Roziere, B., ... & Leather, H. (2023). Large language models for compiler optimization. arXiv preprint arXiv:2309.07062.

Reinforcement Learning from Human Feedback



[source]

LLM Frameworks

ChatGPT Plugins

- Tools designed to assist LLMs on certain tasks.

Wolfram
Access computation, math, curated knowledge & real-time data through Wolfram|Alpha and Wolfram Language.

Expedia
Bring your trip plans to life—get there, stay there, find things to see and do.

Browsing Alpha
An experimental model that knows when and how to browse the internet

Code interpreter Alpha
An experimental ChatGPT model that can use Python, handle uploads and downloads

Other models such as LLaMa have plugins too!

LLM Frameworks

🦜🔗 LangChain Framework

- ❖ Framework for developing applications powered by LLMs.
- ❖ It helps in **context-awareness** and **reasoning**.
  - Concept of “Agent”
  - Memory implementations
  - Interaction with external sources
  - Chains-of-thought
  - Retrieval Augmented Strategies

More information [here](#)!