Radboud University - Nikhef

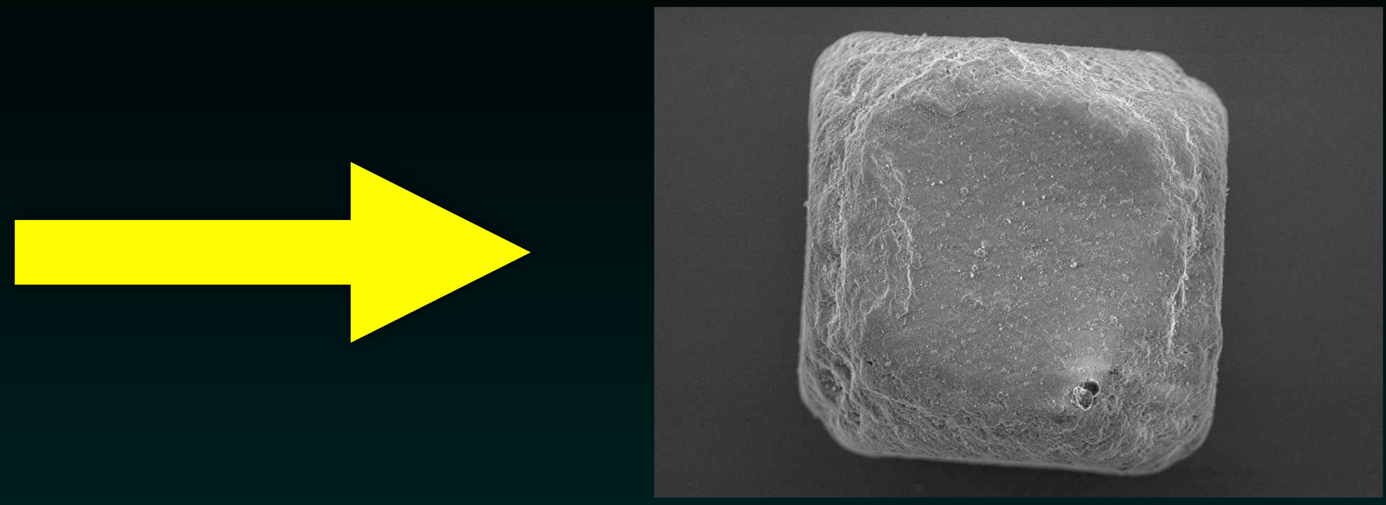
High Energy Physics department - ATLAS team

ChatGPT
as a Coding Assistant

dr. ir. Uraz Odyurt, dr. Geoffrey Gilles

2023-05-15

Say ChatGPT one more time …!

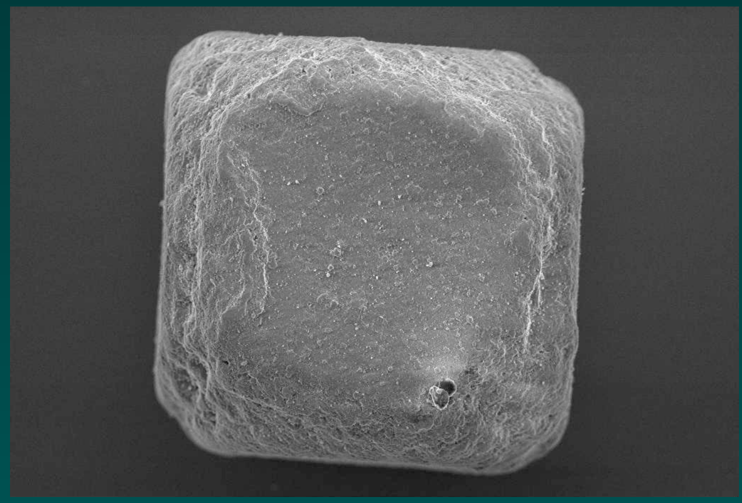


Disclaimer:

- Active use of AI could be controversial …
- For or against, your philosophical opinions are welcome!
- We would like to showcase the possibilities in an unbiased and objective fashion.

“Why should I care?”

- It is being adopted, fast …
- Journals and publishers starting to recognise and regulate its use
- You will fall behind if you don’t consider it (?)



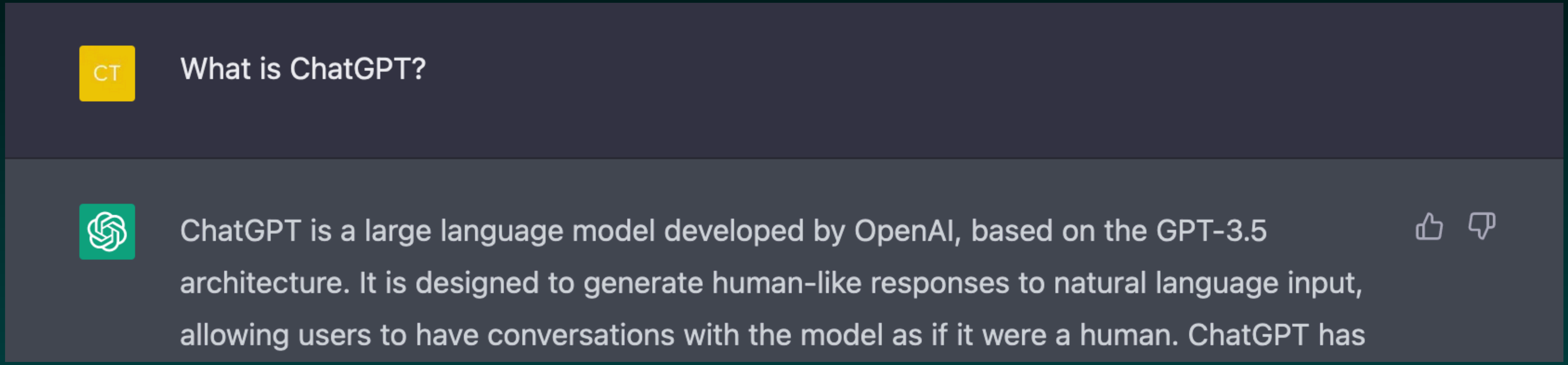
<u>IEEE - Guidelines for Artificial Intelligence (AI)-Generated Text</u>:

“The use of artificial intelligence (AI)–generated text in an article shall be disclosed in the acknowledgements section of any paper submitted to an IEEE Conference or Periodical. The sections of the paper that use AI-generated text shall have a citation to the AI system used to generate the text”.

Stating the obvious …

- What is ChatGPT?

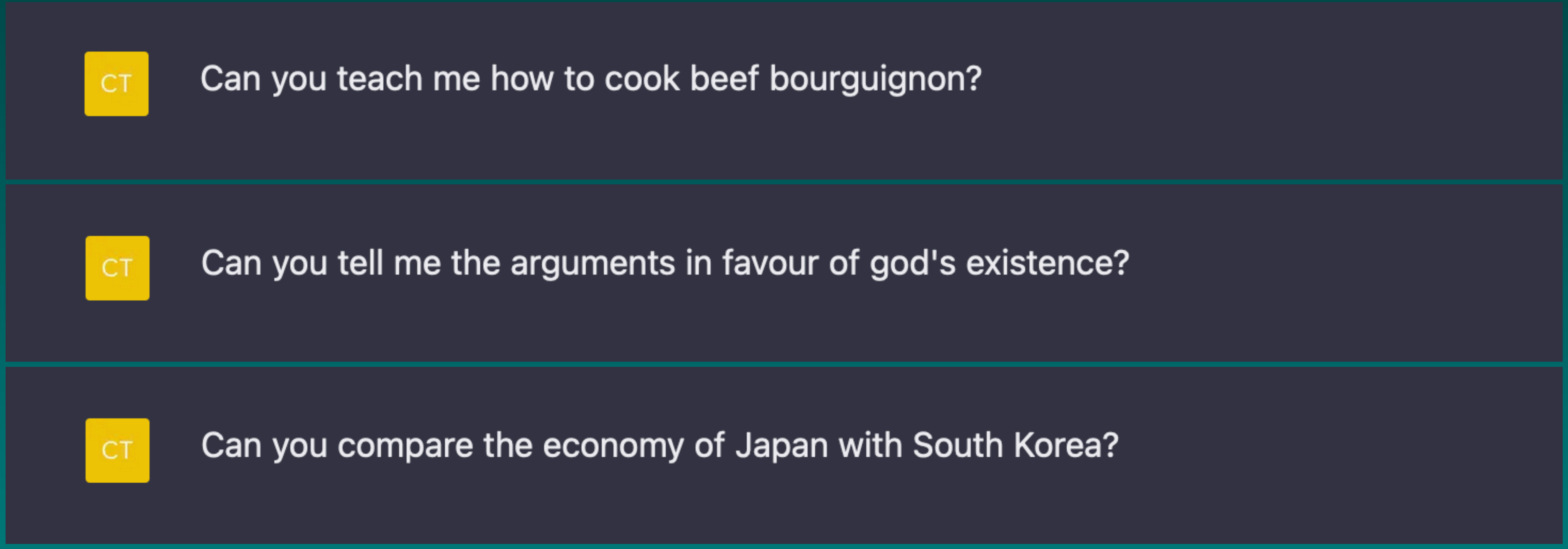
OpenAI



What is ChatGPT?

ChatGPT is a large language model developed by OpenAI, based on the GPT-3.5 architecture. It is designed to generate human-like responses to natural language input, allowing users to have conversations with the model as if it were a human. ChatGPT has

- Language model => Any use-case relevant to languages …
  => Writing
  => Summarising
  => Translation
  => Comparison
  => …
  => **Code generation and tasks related to coding**



Can you teach me how to cook beef bourguignon?

Can you tell me the arguments in favour of god's existence?

Can you compare the economy of Japan with South Korea?

Stating the obvious …

- Natural language and code input:
  => GPT-3.5-turbo -> Input **tokenisation limit is 4096**
  => GPT-4 -> Input **tokenisation limit is 8192**
  => GPT-4-32k -> Input **tokenisation limit is 32768**

- Interaction: **Human-like** responses -> Intended as an assistant

- Let us focus on code generation, as an **assistant for programmers**
  => Code generation
  => Generating documentation
  => Debugging
  => Tutoring and self-learning
  => Alternative ideas, different implementations

Some statistics

- **User acceptance?**
  => 1 million users in 5 days (launched in November 2022)
  => 13 million users daily (January 2023)

- **Costs? (time and money)**
  => \$3 million per month to **run** (estimated - cloud)
  => \$5 million to **train** (estimated - cloud)
  => ~355 years to **train** (single NVIDIA V100 Tensor Core)
  => ~34 days to **train** (1023 NVIDIA A100 Tensor Core)

[source]

A language model you say …

- **Model**: A model is a simplified virtual representation of something real, resembling/predicting its behaviour

- Language **event**: A linguistic unit, text, sentence, token, symbol, …

- Language Model (LM): Given a known **context**, estimate the **probability of following events**
  => Good LM: High probability for correct completions

???

1. His favourite sport, callisthenics, …
2. His favourite sport call, is the nics, …
3. His favourite sport call, is then ics …

A language model you say …

- We cannot have all possible sentences, but the building blocks …
- “Sentence” as a unit is not granular enough.
  => Let’s consider words as **atomic units** …

I	saw	a	raven	flying	over	the	station

$$P(B \cap A) = P(A) \cdot P(B|A)$$

- The joint probability of B and A occurring means the probability of A occurring, multiplied by the probability of B occurring given that **A has occurred** *(context)*

A language model you say …

$$P(A)$$

$$P(B|A) = \frac{P(B \cap A)}{P(A)} \; => P(B \cap A) = P(A) \cdot P(B|A)$$

$$P(C|B \cap A) = \frac{P(C \cap (B \cap A))}{P(B \cap A)} \; => P(C \cap B \cap A) = P(A) \cdot P(B|A) \cdot P(C|B \cap A)$$

B, given A

- The probability of all the words being in one sentence
=> And in the given order -> **Respects the ordering**

$$\begin{aligned} P(e_1, e_2, \ldots, e_n) &= P(e_1) \cdot P(e_2|e_1) \cdot P(e_3|e_1, e_2) \cdot \ldots \cdot P(e_n|e_1, e_2, \ldots, e_{n-1}) \\ &= \prod_{t=1}^{n} P(e_t|e_i : 0 < i < t) \end{aligned}$$

- **A LM lists such probabilities …**

P(* | I saw a raven)
flying = 0.62
walking = 0.27
summer = 0.01

I	saw	a	raven	flying	over	the	station

Access to ChatGPT and cost

- https://chat.openai.com/auth/login
  => Requires registration

- Different tiers of service
  => **Free Research Preview**: Currently running on GPT-3.5
  => **Paid** tier (ChatGPT Plus): Currently running on GPT-4
  (GPT-4 is a different story …)
  => **API** access: Pay per API call scheme

- Embedded in Bing search engine
  => Requires a Microsoft account 🤬

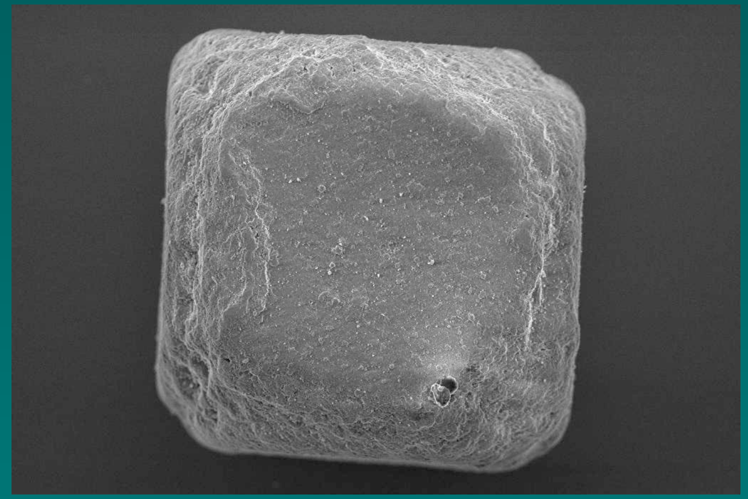
Training data (up to 2021)

- To understand natural language and generate responses
  => Involves NLP techniques, …

- Text data from various sources
  => **Common Crawl**: A large dataset of web pages
  => **BooksCorpus**: A collection of over 11,000 books
  => **Wikipedia**: You know what it is!
  => **OpenWebText**: A curated collection of web pages
  => Stories from **Reddit**: A collection of short stories and comments
  => **English Gigaword**: A large dataset of news articles

Training data (up to 2021)

- Data related to programming languages
  => StackOverflow, GitHub, and other programming forums …
- A variety of technical documentation and tutorials related to programming languages
- Which programming languages?
  => Python, Java, C++, JavaScript, Ruby, and many more …
- Programming languages: **Syntax** + **semantics**
  => **Formal evaluation**



Extensive knowledge of programming languages

- Ability to understand and interpret natural language queries related to programming
  => Syntax
  => Semantics

- Benefitting from the OpenAI “~~**Codex**~~” (deprecated -> GPT-3.5)
  => Demo: Creating a Space Game with OpenAI Codex
  => Separate OpenAI **language model**, designed for code generation
  => Interfaced to a large **curated code database**
  => Millions of **code snippets**
  => **Organised** by language, libraries and frameworks
  => Accompanied by **metadata** on function, inputs and outputs

Example => “GitHub Copilot uses the OpenAI Codex to suggest code and entire functions in real-time.”

As a coding assistant

Error resolution	Code review & optimisation	Algorithm design building blocks	Documentation
• Identifying and resolving coding errors • Guidance - How to fix	• Code analysis and feedback • Optimise code for performance	• Right algorithms • Best libraries and frameworks	• Detailed documentation • Verbose explanations • Simple diagrams
• But inability to solve too complex or overarching issues	• Limited by practical experience and context awareness of the user	• Unable to write extensive code, but rather provide relevant code snippets	• Analytical explanations are generic and follow known knowledge/best practices

Some best practices

Repeat!	• Queries can have different answers • Idea generation by observing different ways of coding • If you didn’t like the answer => Regenerate response!	• Indecision for the user => Can’t decide? Ask for a comparison!
Iterative approach	• An iterative, step-by-step process • Clear action: “Translate this text”, “Write a Python script”, … • More explanation: “Keep writing”, “Keep coding”, …	• Sessions have lasting effects • Start over a fresh session
Provide examples	• Steer the response with examples • Minimal example input	• Sometimes we do not have an example in mind
Concise commands	• State the expected action. • Formatting instructions: “Format in Markdown”, “Limit to 1000 characters”	• Cannot perform tasks requiring intellect/imagination or spontaneous decisions
Tokenise the input	• Surround in backticks/backquotes	• Consider the input tokenisation limit

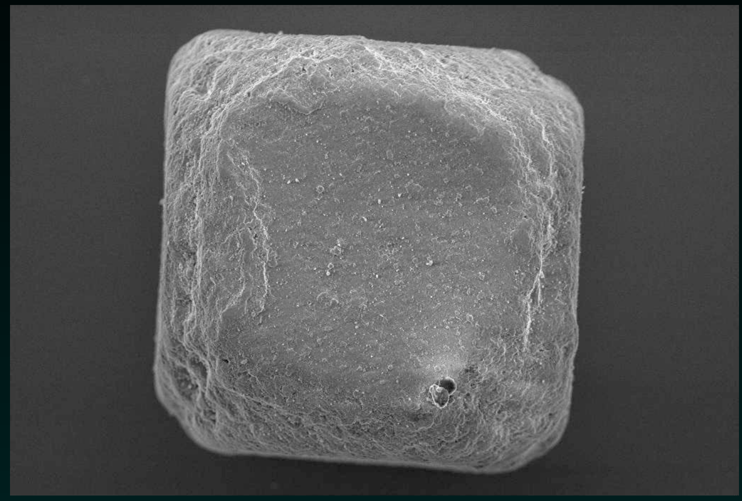
```
`def function(arg1, arg2):
        some code …`
```

Can we trust the output?

Describe 007?

Johnny English
at your service!

Can we trust the output?



- Short answer: No!

- **Validation** is the key
=> You need partial/total **expertise** in the relevant field

- Proper answer: It depends …
=> For languages with an **extensive training corpus**: Yes!
(also programming languages)
=> For **common tasks**: Yes!
(to save time)

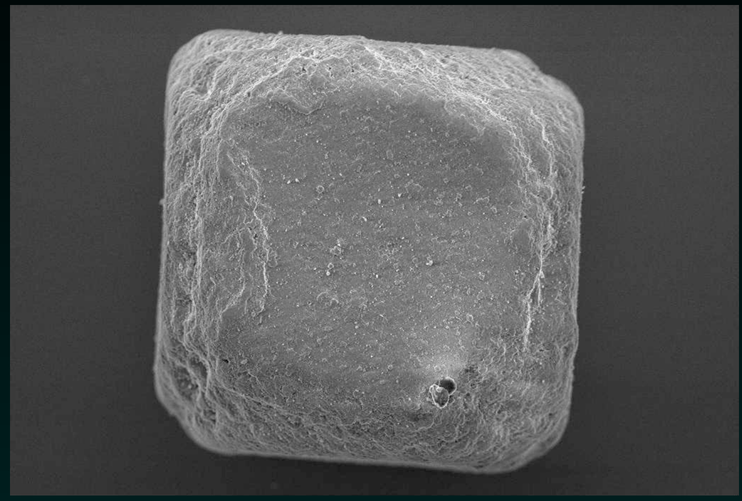
[Towards a Benchmark for Scientific Understanding in Humans and Machines]

How capable is it?

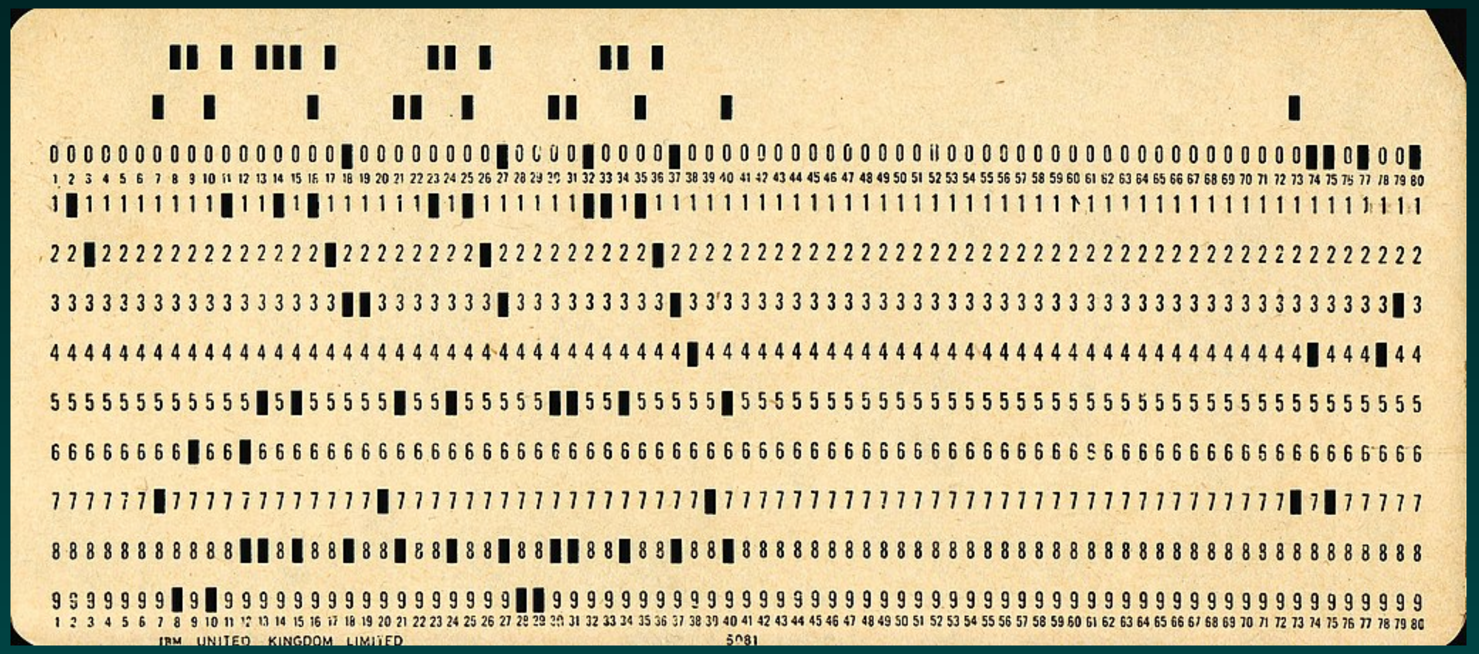
- Get ready to be **impressed**!

- A few examples:
  => HEP task - Madgraph event generation
  => HEP task - ROOT visualisation
  => Specialised coding - Scopus API
  If time allows:
  => Generic coding - Classes in C++
  => Learning - PyTorch, `view` and `reshape`
  => Learning - DataFrame and memory usage
  => Generic coding - Sorting algorithms

Personal takes …

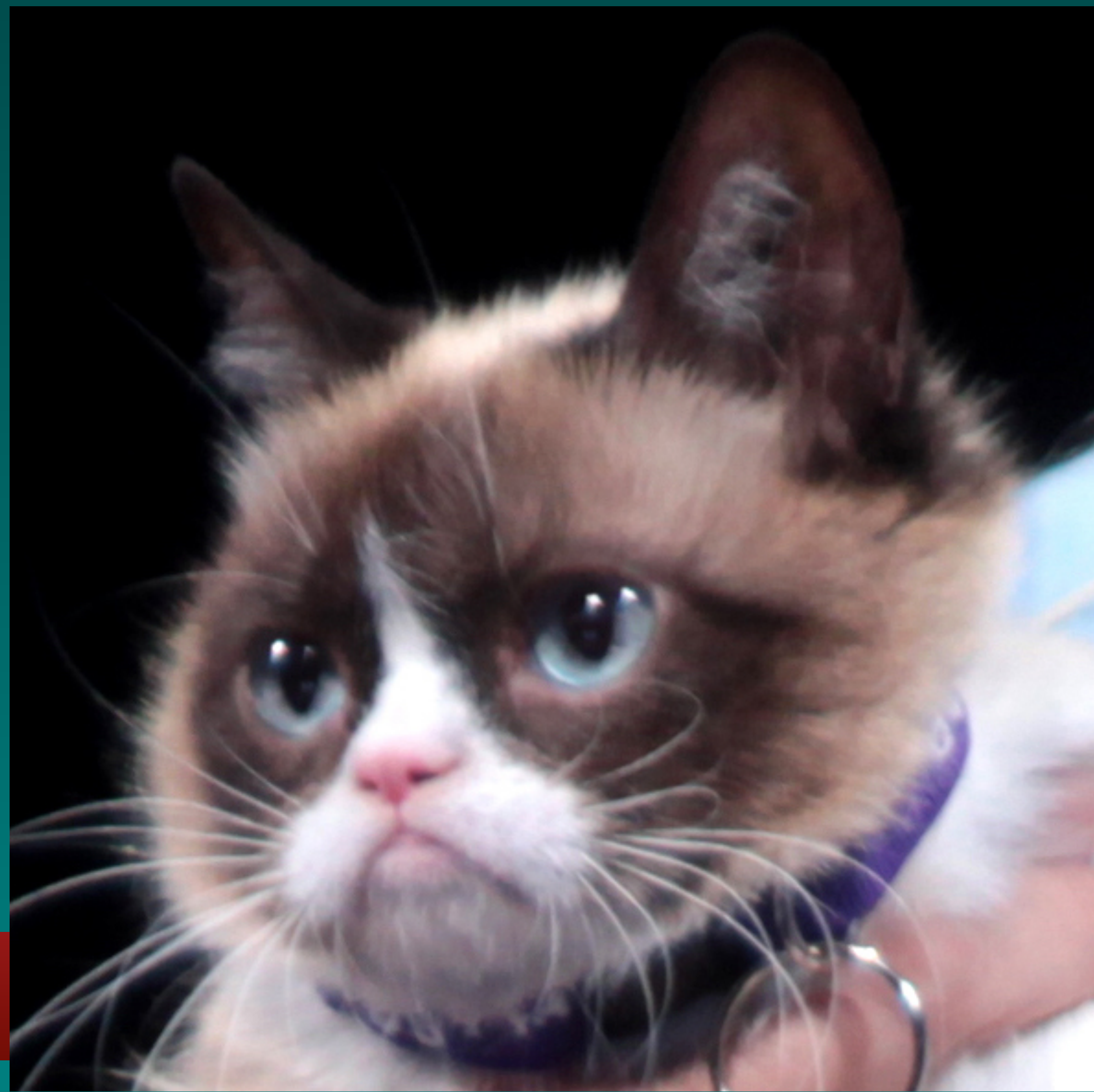


- Is it a good thing to have a machine as our assistant?
  => **YES!** Will be as common as a text editor!



- There are obvious **advantages**
  => Accessible, scalable, improvable, lack of emotions

- People
  => Are not all experts -> we need more experts
  => Get tired -> not available 24/7
  => Have a mind of their own -> order-action style of interaction
  => Are volatile -> grumpy, angry, sad, moody, bored, impatient, overwhelmed, underwhelmed, …



Let’s try it out … !

CT
python
Copy code
Regenerate response

Tell me which sorting algorithm is the fastest and provide the C++ implementation of it?

The fastest sorting algorithm depends on the specific use case and the characteristics of the data being sorted. Generally, quicksort is considered one of the fastest sorting algorithms for most use cases, with an average time complexity of O(n log n). However, it has a worst-case time complexity of O(n^2), which can occur when the input is already sorted or nearly sorted. Other sorting algorithms such as merge sort and heapsort also have an average time complexity of O(n log n), but their worst-case time complexity is better than quicksort.

Here is a C++ implementation of quicksort:

Now rewrite the sort in Python.

Here's an implementation of quicksort in Python:

```python
def quicksort(arr):
    if len(arr) <= 1:
        return arr
    else:
        pivot = arr[len(arr)//2]
        left = [x for x in arr if x < pivot]
        middle = [x for x in arr if x == pivot]
        right = [x for x in arr if x > pivot]
        return quicksort(left) + middle + quicksort(right)

arr = [5, 2, 9, 3, 7, 4, 8, 1, 6]
sorted_arr = quicksort(arr)
print(sorted_arr)
```

This implementation is a bit different from the C++ implementation. Instead of using indices to partition the array, it uses list comprehension to create three separate lists for elements less than, equal to, and greater than the pivot element. These lists are then concatenated together in the correct order to produce the sorted array.

The function takes in a list `arr` and recursively calls itself on the left and right partitions until the entire list is sorted. The base case for the recursion is when the length of the list is less than or equal to 1, indicating that it contains only one element.

```
#include <iostream>

class Car {
    public:
        // Constructor
        Car(std::string m

        // Member functio
        void start() {
            std::cout << "Starting the " << make_ << " "
        }
        void stop() {
            std::cout << "Stopping the " << make_ << " "
        }

    private:
        // Data members
        std::string make_;
        std::string model_;
        int year_;
};

class Lorry : public Car {
```

Can you explain with simple examples how pytorch view interacts with tensors?

Sure, I'd be happy to explain how PyTorch's `view` function interacts with tensors using some simple examples.

The `view` function in PyTorch is used to reshape a tensor without changing its underlying data. It returns a new tensor with the same data but with a different shape. Here are some examples to demonstrate its usage:

```python
import torch

# create a 1D tensor
x = torch.tensor([1, 2, 3, 4, 5, 6])
print(x)
# Output: tensor([1, 2, 3, 4, 5, 6])

# reshape the tensor into a 2D tensor with 2 rows and 3 columns
y = x.view(2, 3)
print(y)
# Output: tensor([[1, 2, 3],
#                 [4, 5, 6]])

# reshape the tensor into a 3D tensor with 2 channels, 1 row, and 3 columns
z = x.view(2, 1, 3)
print(z)
# Output: tensor([[[1, 2, 3]],
#                 [[4, 5, 6]]])
```

nd a s

API to send a s

query and receives back the resulting publications' metadata:

```python
import requests
import json

# Set the API key and search query parameters
api_key = 'YOUR_API_KEY'
query = 'TITLE-ABS-KEY("data mining") AND PUBYEAR AFT 2018'

# Define the API endpoint and request parameters
url = 'https://api.elsevier.com/content/search/scopus'
params = {'apiKey': api_key, 'query': query, 'field': 'dc:identifie

# Send the request to the Scopus Search API and get the response
response = requests.get(url, params=params)

# Parse the response and extract the publications' metadata
if response.status_code == 200:
    data = json.loads(response.text)
    for item in data['search-results']['entry']:
        identifier = item['dc:identifier']
        title = item['dc:title']
        journal = item['prism:publicationName']
        date = item['prism:coverDate']
```

Can you write the generated data directly to the CSV file, not using a Data Frame?

Yes, you can write the generated data directly to a CSV file without using a DataFrame. Here's an example code that generates a list of tuples containing the same data as in the previous example, and writes it directly to a CSV file:

```python
import csv
import numpy as np

# Generate dummy data
names = ['Alice', 'Bob', 'Charlie', 'David', 'Emily']
ages = np.random.randint(low=20, high=50, size=len(names))
cities = ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Miami']

# Write data to CSV file
with open('dummy_data.csv', mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['Name', 'Age', 'City'])
    for i in range(len(names)):
        writer.writerow([names[i], ages[i], cities[i]])
```

```
process = MG.MadGraph5Interface(proc_card='p p > h > gamma gamma',)

# Set the process energy and number of events to generate
process.run_cfg['run_mode'] = 0  # "Run" mode for event generation
process.run_cfg['nb_event'] = 1000  # Number of events to generate
process.run_cfg['energy'] = 13000  # Center-of-mass energy in GeV

# Generate events and save them to a file
process.run_cmd()
```

Considering the above two snippets, which one has better performance?

Generally, writing data directly to a CSV file using the `csv` module can be more performant than creating a DataFrame and then writing it to a CSV file using pandas. This is because pandas is a powerful and flexible library that provides a lot of functionality, but that comes at a cost of performance overhead.

However, the performance difference between the two methods may not be significant for small datasets. In fact, for small datasets, the overhead of using pandas may be negligible, and the simpler and more concise syntax of using pandas may be more appealing.

In summary, if you're dealing with a large dataset, writing directly to a CSV file using the `csv` module may provide better performance. On the other hand, if you're working with a small dataset, the performance difference may not be significant, and you may prefer to use the simpler and more concise syntax of pandas.

We happy?

Thank you!
Questions?