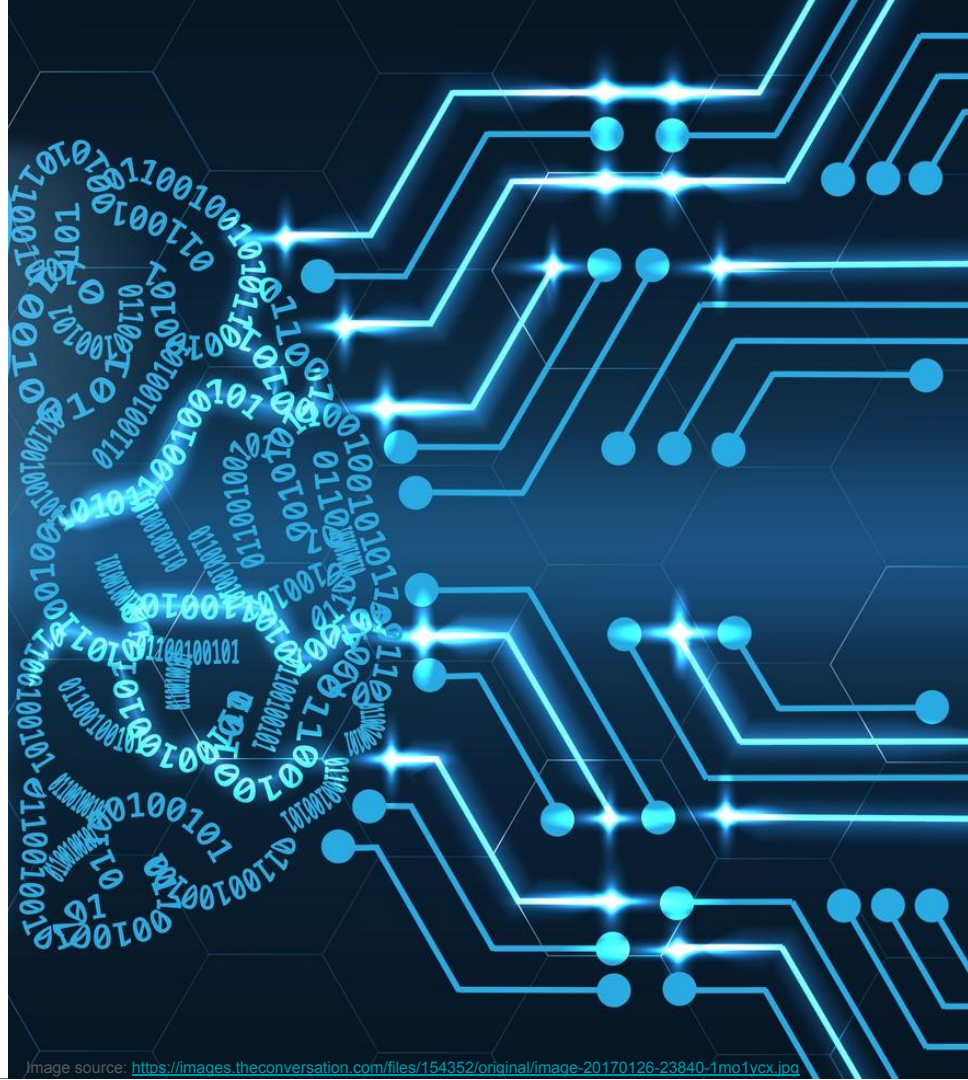


# Understanding the LLM landscape

An early 2025 overview

Lize Pirenne and Prof. Damien Ernst



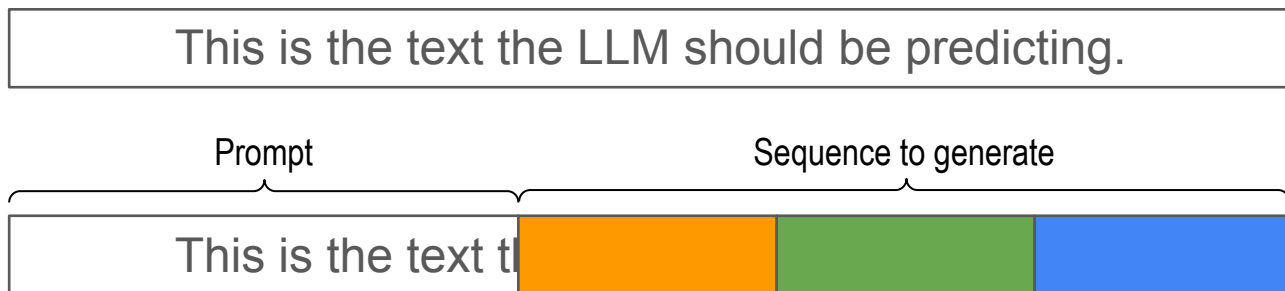
Understanding the LLM landscape

## **Chapter 1**

**What is a Large Language Model?**

# What is an (autoregressive) Large Language Model?

A Large Language Model (LLM) generates a sequence (the response/completion) that is likely to follow another one (the prompt).



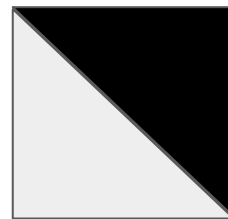
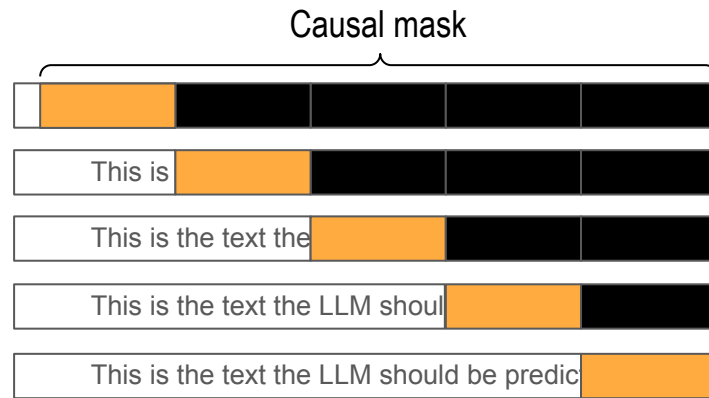
The autoregressive model defining the LLM iteratively gives a probability distribution over a random variable  $\square$  (the next token) conditioned on the whole past. Its realisation  $\square$  is then used for the next iteration.

# Learning for autoregressive models

The data for learning is made of causally masked sequences where the future tokens are hidden. The resulting autoregressive model is the one that best predicts this data on average.

More formally, the learning process minimises the cross entropy between the output distribution  $P(\square|\square)$  and the correct token (a Dirac delta function distribution).

High-probability completions are seen as more desirable because they represent the data accurately.



It looks triangular in a big matrix.

# A deeper look into the probability of a completion

The probability of the entire completion can be computed with the chain rule:

$$P(\square, \square, \square | \square) = P(\square | \square) P(\square | \square, \square) P(\square | \square, \square, \square).$$



An autoregressive LLM gives:

- $P(\square | \square)$ ;
- then  $P(\square | \square, \square)$ ;
- and then  $P(\square | \square, \square, \square)$ .

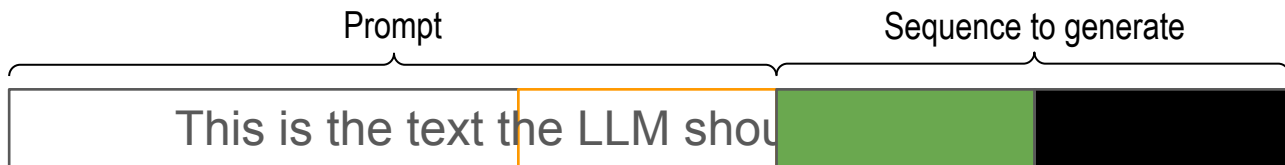
# Example of autoregression (1/3)

First, we compute the distribution for the first token  $P(\square|\square)$  using the prompt  $\square$ .



## Example of autoregression (2/3)

After having drawn a sample  $\square$  from the first distribution  $P(\square|\square)$ , we compute the distribution for the second token  $P(\square|\square, \square)$ .



## Example of autoregression (3/3)

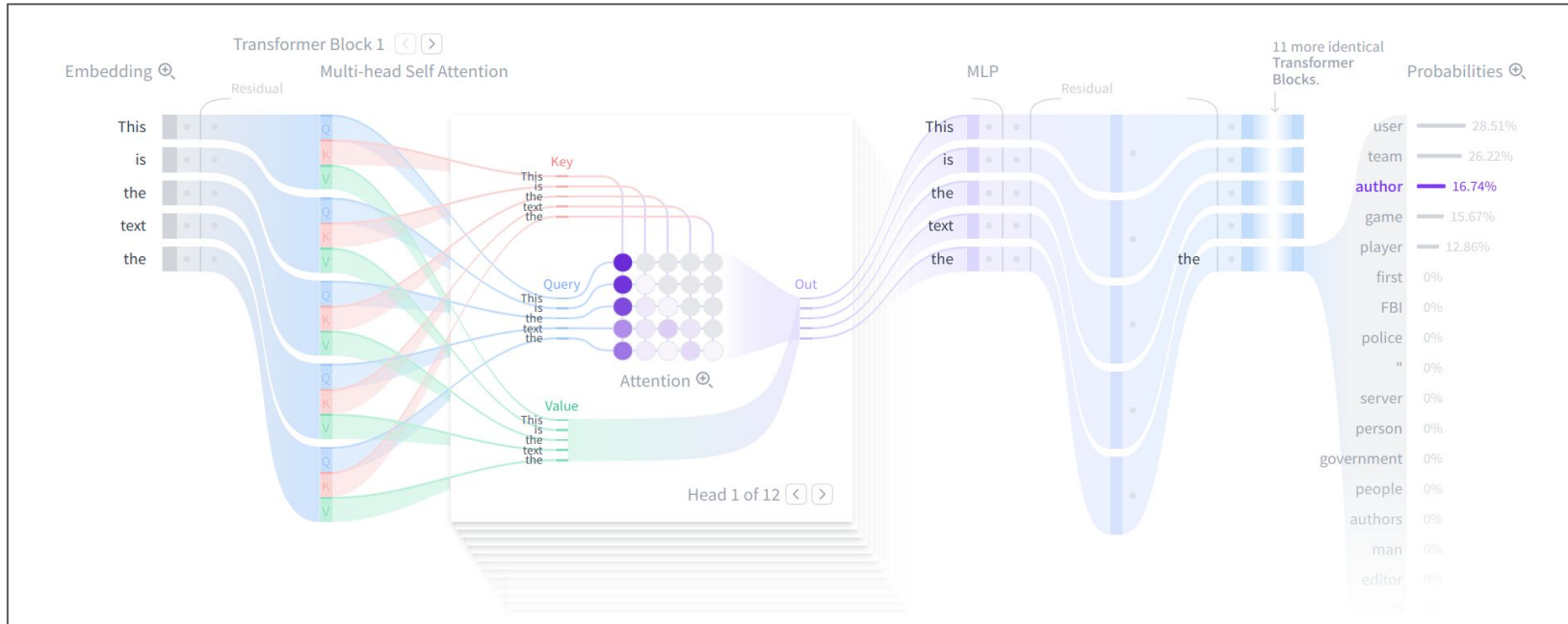
Then we draw the second sample  $\square$  from the second distribution  $P(\square|\square,\square)$  and, finally, we compute the distribution for last token  $P(\square|\square,\square,\square)$ .





# Architecture

The most common architecture used to build LLMs is called a decoder-only transformer.



Understanding the LLM landscape

## **Chapter 2**

### **LLM as a business**

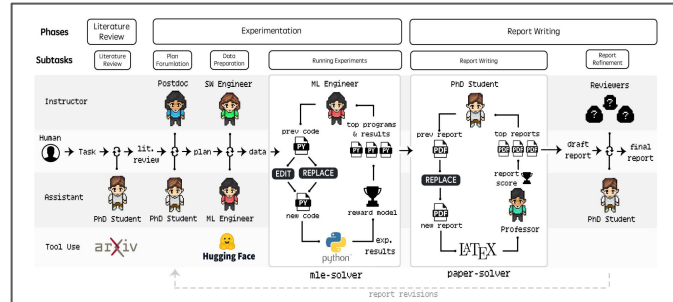
# Uses of LLMs

Such a simple process allows for amazing applications, for example:

## Chatbots



## AI-assisted research



## Content creation with AI


































# LLM provider, a new business

LLM providers are businesses that host LLMs and centralise the requests of their customers.

They often used proprietary LLMs that they have trained at huge cost but some of them also use open-source LLM models.



Those open-source models can in principle be run by anyone, but the cost of hardware is often prohibitive. Indeed, running your model locally does not allow you to amortise it over a large number of requests, as LLM providers do; they also benefit from economy of scale for building their IT infrastructure.

 OpenAI	 Mistral
 Microsoft Azure	 Hyperbolic
 DeepSeek	 Amazon Bedrock
 Groq	 FriendliAI
 Together.ai	 Anthropic
 Perplexity	 Google
 Lambda Labs	 Fireworks
 Cerebras	 Cohere
 Upstage	 Simplismart
 Deepinfra	 Replicate
 Nebius	 MiniMax
 CentML	 kluster.ai
 Reka AI	 Databricks
 SambaNova	 xAI
 Alibaba Cloud	 Novita
 AI21 Labs	

# Provider prices

Prices of providers are given per (million) token.

Typical cost for a provider, here Lambda, hosting a deepseek model:

API PROVIDER <small>↑↓</small>	MODEL <small>↑↓</small>	BLENDED <small>↑↓</small> <i>USD/1M Tokens</i>	MEDIAN <small>↑↓</small> <i>Tokens/s</i>	MEDIAN <small>↑↓</small> <i>First Chunk (s)</i>	TOTAL <small>↑↓</small> <i>Response (s)</i>
 <b>Lambda</b>	 <b>DeepSeek R1</b>	\$0.95	40.1	0.54	71.51

We see that we pay only \$0.95 per 1 million tokens. The model produces around 40.1 tokens per second. It needs 0.54 seconds to output the first token.

# Running DeepSeek R1 locally

To run the same DeepSeek R1 model locally and produce the same number of tokens per second with the same latency, you would need  $20 \times$  A100 80GB.

This would correspond to a cost of:


$$20 \times \$17,549.95 = \$333,449.05.$$

The DeepSeek R1 model run by Lambda could generate around 350,000 million tokens for this price.



# Little models for little budgets

Some people want to have their own LLM for privacy reasons (for example) without investing in expensive hardware. Smaller models, only slightly poorer performance, are suitable for them.

Some of these models can even run on your smartphone and dedicated apps have been developed for this, such as the PocketPal AI ( PocketPal AI LLM Ventures))).

	Model Variant	Parameters	Unified Memory
One phone*	DeepSeek-R1-Distill-Qwen-1.5B	1.5B	3.9 GB
	DeepSeek-R1-Distill-Qwen-7B	7B	18 GB
One Nvidia 4090	DeepSeek-R1-Distill-Llama-8B	8B	21 GB
One A100 40G	DeepSeek-R1-Distill-Qwen-14B	14B	36 GB
	DeepSeek-R1-Distill-Qwen-32B	32B	82 GB
	DeepSeek-R1-Distill-Llama-70B	70B	181 GB
The one offered by providers	DeepSeek-R1-Zero-671B	671B	1,543 GB

# Compression techniques for making models simpler

Models can be compressed to around a fourth of their original size (almost) without modifying the output using various techniques such as “quantisation”.

This technique represents the numbers corresponding to the parameters of the model with a smaller number of bits.

Quantised:			Original:
Model Variant	Parameters	Unified Memory	Unified Memory
DeepSeek-R1-Distill-Qwen-1.5B	1.5B	1 GB	3.9 GB
DeepSeek-R1-Distill-Qwen-7B	7B	4.5 GB	18 GB
DeepSeek-R1-Distill-Llama-8B	8B	5 GB	21 GB



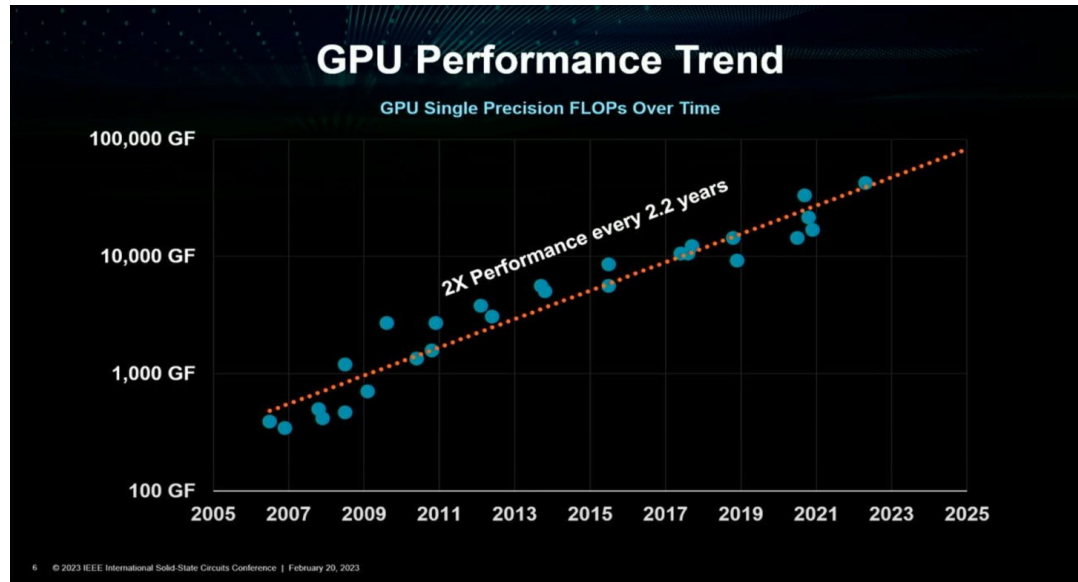
Understanding the LLM landscape

# **Chapter 3**

## **Growth of LLMs**

# How did LLM become so good?

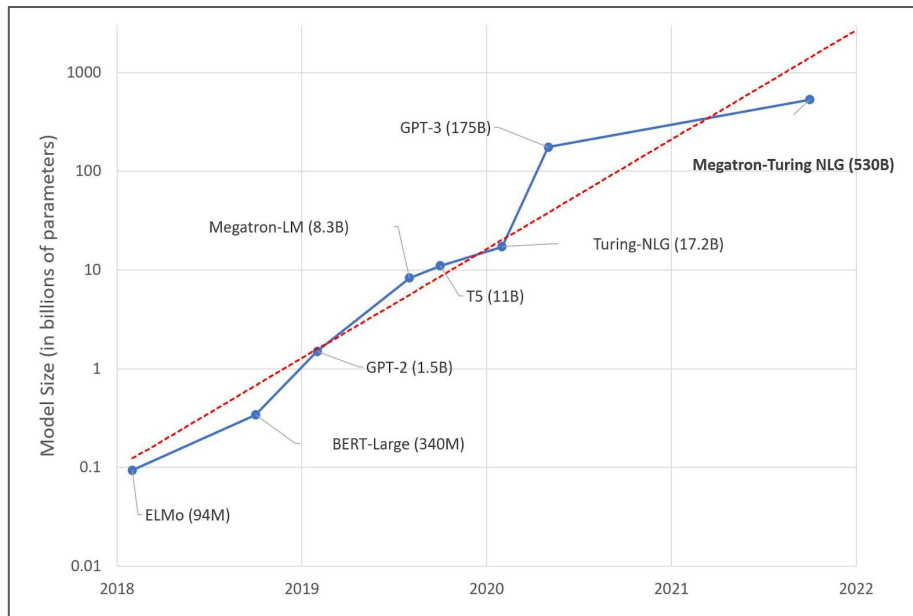
The excellent performances of today's LLMs would not have been possible without the tremendous growth in the computational capabilities of GPUs.



# Scaling for the win

Language Models demonstrated scaling laws: “bigger is better” with no sign of stopping

The switching point in terms of performances happened with GPT-3, which is ten-times bigger and used much more data than previous models. It paid off.



# The realm of open-source models

ChatGPT is a good model, but it was closed source.

After its release, open-source models quickly caught up. Big software companies, like Meta or Google made their models available for anyone to download and run.

Examples of well-known open-source models:

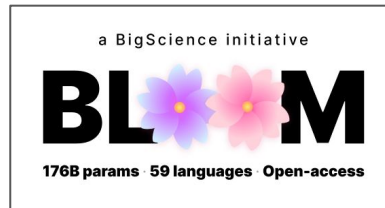
LLaMa by Meta



PaLM 2 by Google



BLOOM by HuggingFace



# Benefits of releasing open-source models

There were three incentives for big corporations to release their models as open source:

- (i) Their name would be on everyone's lips;
- (ii) The ecosystem would revolve around their standards;
- (iii) Researchers could spot weak points and fix them for free.

# Evaluating the quality of an LLM: benchmarks

Datasets made for evaluating the quality of LLMs are called benchmarks. One of the first widely used benchmarks was GMSK8.

For this benchmark, the prompt defines a Problem to be solved. For every problem, the benchmark gives the Solution scheme and the answer. The LLM is judged on its capacity to provide the Final answer.

**Problem:** Beth bakes 4, 2 dozen batches of cookies in a week. If these cookies are shared amongst 16 people equally, how many cookies does each person consume?

**Solution scheme:** Beth bakes 4 2 dozen batches of cookies for a total of  $4*2 = \ll 4*2=8 \gg$  8 dozen cookies. There are 12 cookies in a dozen and she makes 8 dozen cookies for a total of  $12*8 = \ll 12*8=96 \gg$  96 cookies. She splits the 96 cookies equally amongst 16 people so they each eat  $96/16 = \ll 96/16=6 \gg$  6 cookies.

**Answer:** 6.

<https://arxiv.org/pdf/2110.14168v1.pdf>

The accuracy of the LLM is computed simply by checking that we have a complete match of the answer.

# Stochastic parrots

A well-thought quote about LLMs being stochastic parrots:

*“A system for haphazardly stitching together sequences of linguistic forms it has observed in its vast training data, according to probabilistic information about how they combine, but without any reference to meaning.”*

Emily M. Bender et al.



In order to be more performant than big stochastic parrots, as GPT-3 is for example, we need something more than textual data and bigger models. One strategy is to make better use of LLM properties.

Understanding the LLM landscape

## **Chapter 4**

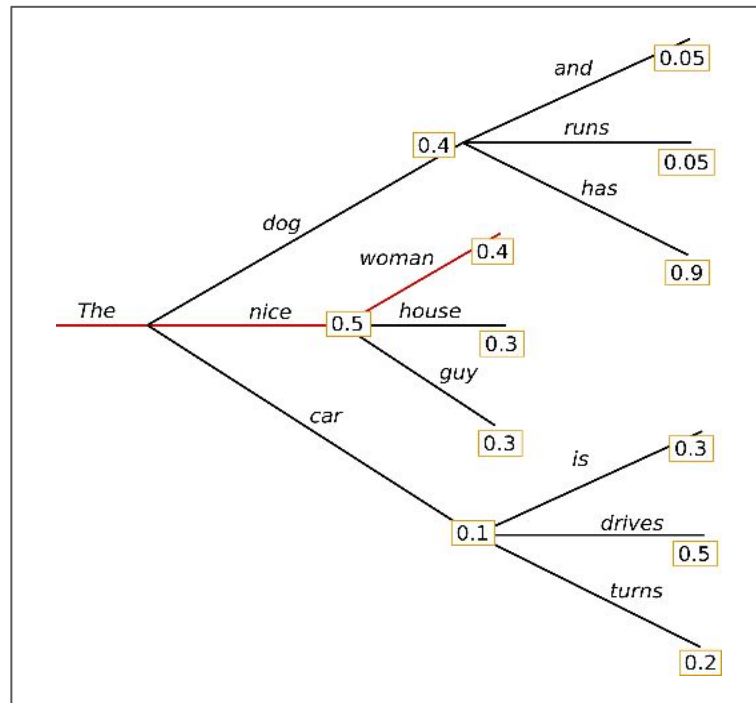
# **Making better use of LLMs properties**



# Better use 1: Searching for the best completion

The ensemble of all completions can be represented as a tree where each child of a node is possible next token with its probability. The probability of a path in the tree gives the probability of a completion.

One possible solution for making a better use of existing LLM models would be to output the path with the highest probability and not, as it is the case now, a path generated by sampling the autoregressive model.



# A computational issue

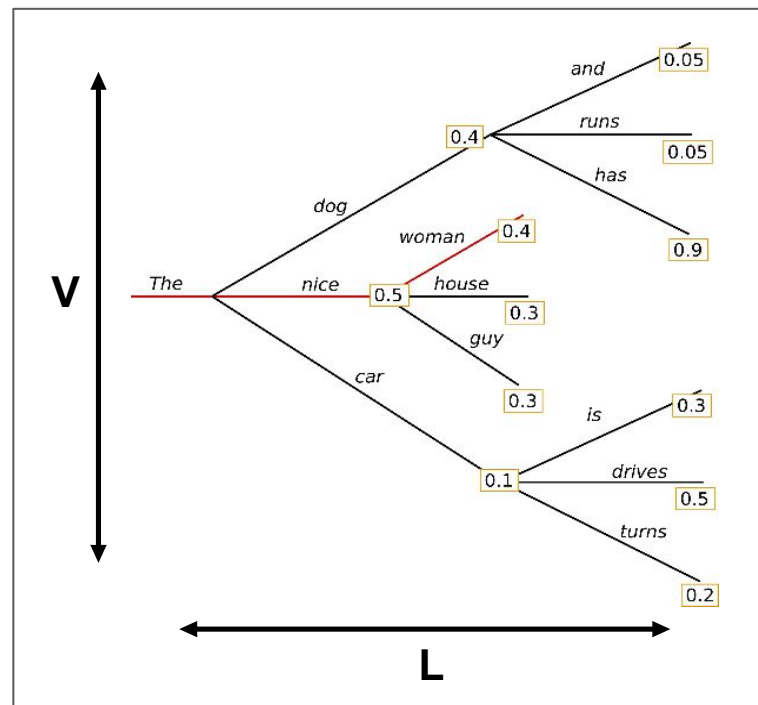
To find the maximum of the auto-regression tree, we have in principle to explore all possible nodes.

This is computationally out of reach.

## Example:

Deepseek-R1 has a vocabulary size  $V$  of around 131,072 tokens. The average response length  $L$  is around 10,000 tokens.

Thus, as first approximation the number of paths in the tree is  $131,072^{10,000}$ .

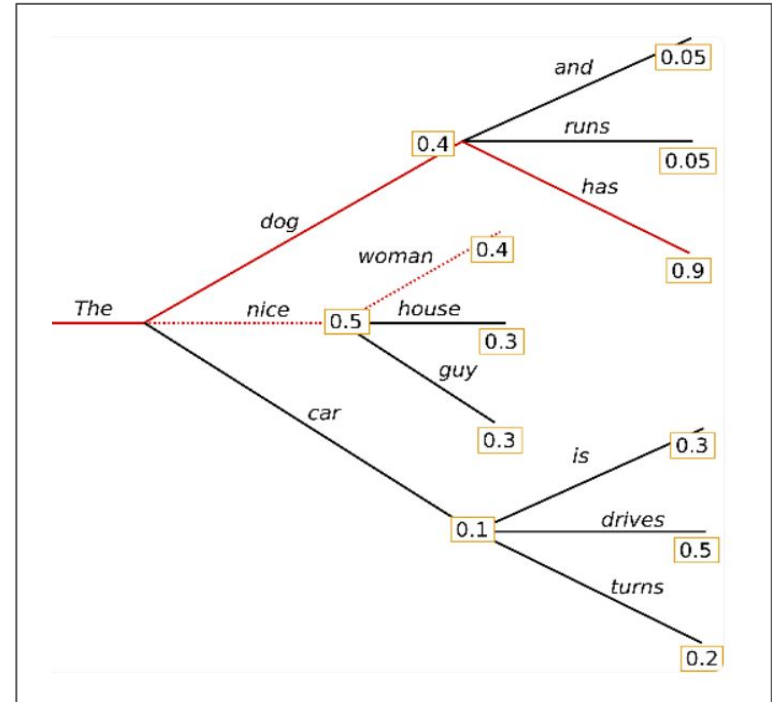


## Exploring the tree efficiently

Many scientists have developed algorithms for efficiently exploring this tree in order to rapidly identify a high-probability path.

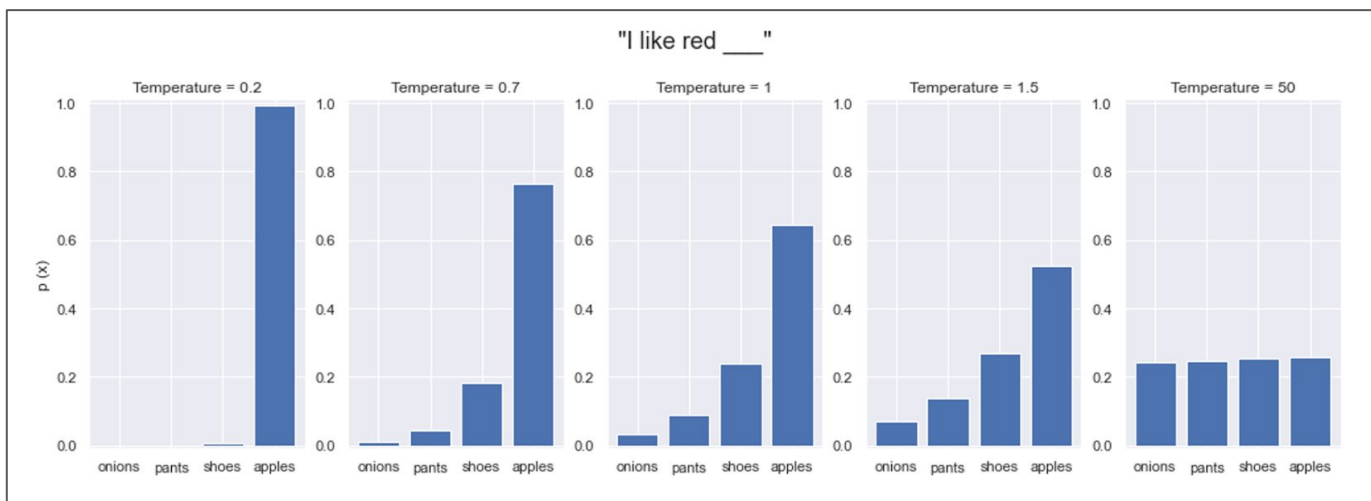
One of the most common strategies is to grow only a limited number of paths by favouring branches having a high probability.

In this example, only the red-coloured paths are explored.



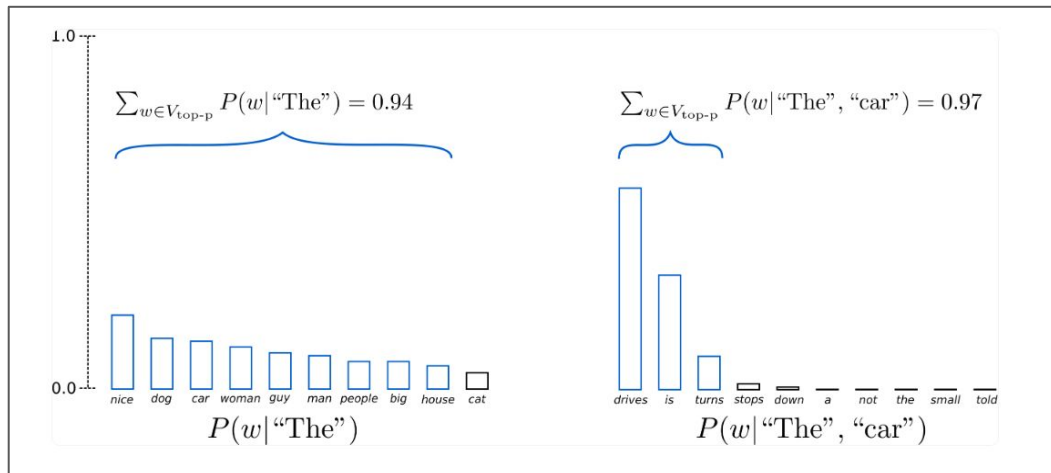
# Better use 2: Fostering creativity with temperature

Some use-cases require creativity, as is the case when writing an original book. In such cases, we may want to diverge from the original data distribution. This can be done by changing the temperature parameter that tweaks the distribution defined by the LLM towards a more uniform one.



# Better use 3: Limiting unlikely samples with nucleus sampling

We may want creativity, but we also want to stay in the realm of possible completions. Many tokens have overestimated probabilities (especially with high temperature). We can choose to only include those who contribute the most (top-k sampling), or up to a certain total percentage. This is called nucleus sampling.



# Better use 4: Prompting better with Chain-of-Thoughts

On the right two outputs of LLM on a question of the GSMK8 benchmark. The first is asked with a straightforward prompt but the question does not give a correct answer.

Adding a simple line “*Let us think step by step*” to the beginning of the LLM’s completion gives the correct answer.

The LLM can perform better if asked to come with the reasoning steps leading to the answer, that is to develop a Chain-of-Thoughts.

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

The answer is 27. ❌

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have? *Let's think step by step.*

Model Output

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. ✅

# Test-time compute

The methods developed in this chapter are grouped under the name test-time compute since they leverage computation time after the training\* is over.

Test-time computing is favoured with smaller models as it can be seen as an approach to offset the loss of performances associated with model size reduction.

Choosing the right trade-off between training and test-time compute depends also on the usage. But notice that all compute is not equivalent.

## Takeaways for exchanging pretraining and test-time compute

Test-time and pretraining compute are not 1-to-1 “exchangeable”. On easy and medium questions, which are within a model’s capabilities, or in settings with small inference requirement, test-time compute can easily cover up for additional pretraining. However, on challenging questions which are outside a given base model’s capabilities or under higher inference requirement, pretraining is likely more effective for improving performance.

Quote from: <https://arxiv.org/abs/2408.03314>

\*The entire training phase is usually done in three phases: pretraining, fine tuning and reinforcement learning.

Understanding the LLM landscape

## **Chapter 5**

# **Emulating intelligence**



# Road to intelligence: diversified inputs

To start building intelligence, we want models to have a more accurate view of how concepts relate to each other. Let us take the following example:

*“An apple falls under the effect of gravity.”*

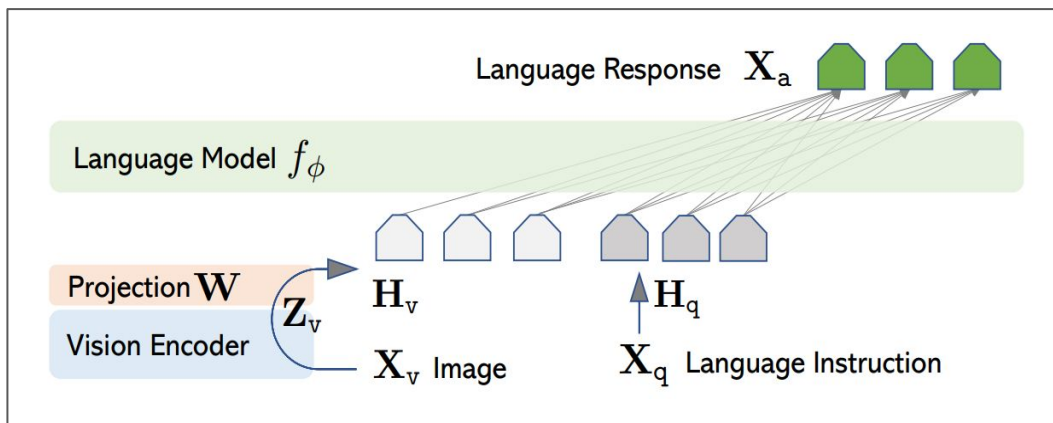
The object “apple” can be influenced by the force “gravity” creating the result of “falling”. There are clearly a lot of concepts hidden behind this sentence.

Would it not be easier for a machine to understand them if it had access to a video of the apple falling and not only the text? This suggests that as a first step on the road to intelligence to have LLMs that can process multiple modalities.

# Adding senses to LLMs: multimodality

We now see LLMs appearing that can understand many modalities.

The strategy used to achieve this is quite simple: a mapping of all the different inputs into the same space, the one whose elements are further processed by the LLM. This can be done efficiently by training a little module that will translate (project) the different inputs into this space.



# Computational challenges related to multimodal LLMs

Multimodal LLMs (MLLMs) process very large inputs made among others of audio-visual signals, and this requires much more hardware capabilities.

The research community has had to address these computational challenges. It proposed two main improvements:

- (i) Multi Layer Attention: Reusing the same blocks for different layers of the language model  $f_\phi \rightarrow \sim 98.2\%$  more efficient ( $56.9 \times$  compression factor);
- (ii) Flash attention: Reordering the matrix multiplications to take advantage of hardware infrastructure.

## An example of multimodality in action



# Are big multimodal models intelligent?

The goal of MLLMs is still the same as regular LLMs: Best prediction of what should follow a prompt given all input data in the multimodal prompt and its prior beliefs from its training.

MLLMs are still stochastic parrots, but their generations are a lot easier due to the enormous conditioning provided by the prompt.

*Multimodality is good, but it is not sufficient to have general intelligence.*



# Road to intelligence: adaptation

Humans are masters of adaptation. We want to give that capacity to LLMs too.

Let us define a setting in which an LLM would learn from an evaluation of its performances through a numerical score/reward:

- An LLM gives an answer to a prompt;
- A human or computer software quantifies the quality of this answer by giving a numerical score/reward to it;
- The LLM modifies its behaviour such that it would get better rewards next time.

## Question:

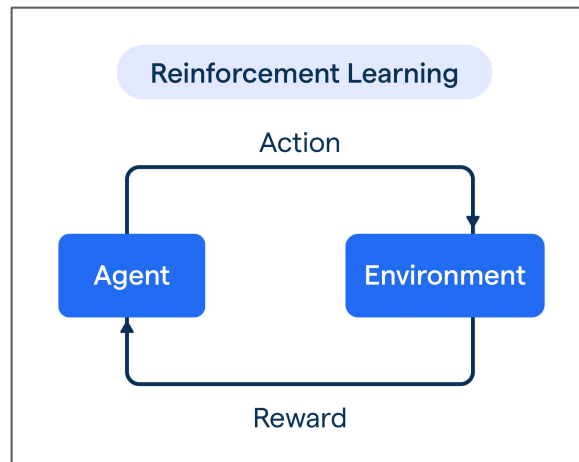
How can the LLM indeed modify its behaviour in such a setting?

# Reinforcement learning algorithms for adaptation

Reinforcement Learning (RL) is a subfield of Artificial Intelligence focusing on the design of algorithms for making an agent learn from its environment through a numerical reward signals that scores its behaviour.

In a typical RL setting, an agent applies an action to its environment and in return gets a reward and observations of the environment.

In our case, the LLM is the agent's policy generating the token. The observation signals are the tokens. The environment can be a conversation\*, a coding space... and the rewards score the behaviour of the LLM.



\*When humans are giving the feedback, it is called RLHF (RL from Human Feedback).

# Are models using intelligent human feedback?

Not necessarily. They can still be seen as parrots that have learnt to please humans instead of copying them. Indeed, they have been training with the goal of producing a high-probability completion, though the highest probability sentence reflects human preferences.

Additionally, a common phenomenon happening when training with reward models is reward hacking: *The model finds a strategy to maximise the rewards in training and this strategy does not necessarily generalise to problems not seen during training\**.

This highlights a shortcoming in intelligence of LLMs trained in a RL setting.

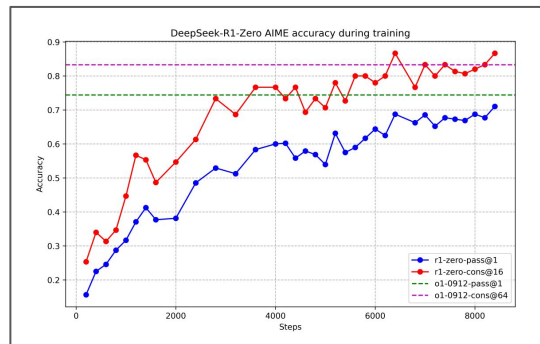
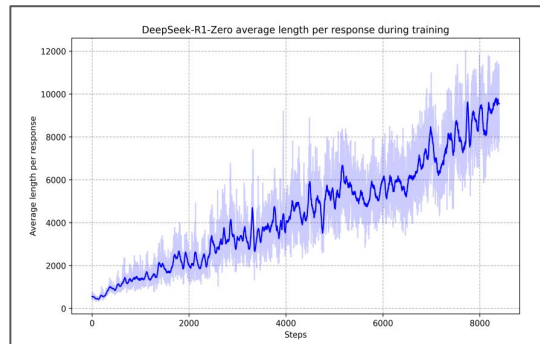
\*We note that one way to mitigate reward hacking is to force models to stay close to the original one by adding a KL divergence term.



# Addressing this shortcoming using rewards with “Chain-of-thought” flavour

By just adding to the rewards terms that scores not only the final answer, but also the presence of reasoning leading to the answer, better performances can be achieved.

In the first image, we see that the reasoning length increases with training when using such rewards. In the second image, we see two metrics for evaluating the quality of the answer growing with the RL training time.



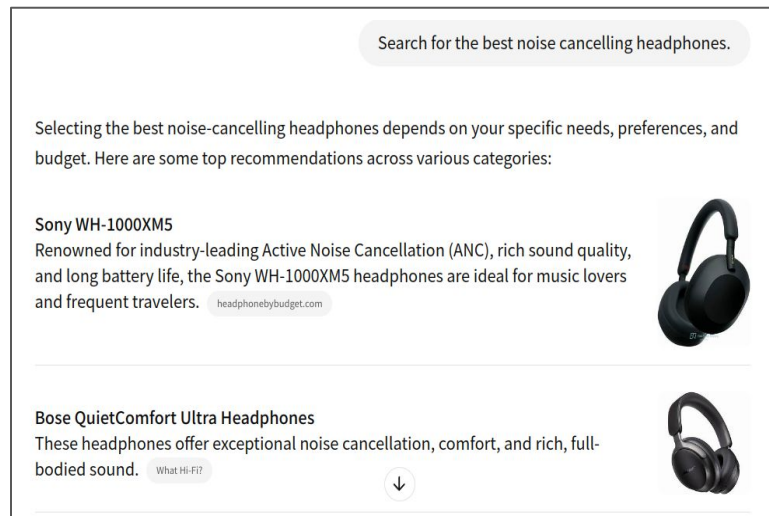
# Road to intelligence: LLMs interacting with tools

An LLM can use external tools when appropriate. This can greatly empower them! The call to external tools is simply done by generating the right text for APIs.

An example of an LLM using tools to better answer a question.

This is done in several steps:

- (i) Select “Google Search” as first tool to use;
- (ii) Make an API call to this tool;
- (iii) Summarise the information gained;
- (iv) Select the “Format answer” tool to display images in a nice way.



Understanding the LLM landscape

## **Chapter 6**

**Finding the rationale  
(My research)**

# Explaining LLM answers in Question Answering (QA)

In closed-context QA, there is often a requirement to find the elements of the context that were used to produce the answer. These elements are collectively called the rationale. This is particularly desirable for having trust in the answer of the LLMs and sourcing the information used.

My research focuses on developing techniques to find rationales.

One example is on the right, the rationale found by one of the technique I have used is the text whose background is coloured in black.

```
Layer 5, Head 4
### Question: What is Sauvignon blanc?

### Context: Sauvignon blanc is a green-skinned grape variety that originates from the city of
Bordeaux in France. The grape most likely gets its name from the French words sauvage ("wild") and
blanc ("white") due to its early origins as an indigenous grape in South West France. It is
possibly a descendant of Savagnin. Sauvignon blanc is planted in many of the world's wine regions,
producing a crisp, dry, and refreshing white varietal wine. The grape is also a component of the
famous dessert wines from Sauternes and Barsac. Sauvignon blanc is widely cultivated in France,
Chile, Romania, Canada, Australia, New Zealand, South Africa, Bulgaria, the states of Oregon,
Washington, and California in the US. Some New World Sauvignon blancs, particularly from
California, may also be called "Fumé Blanc", a marketing term coined by Robert Mondavi in
reference to Pouilly-Fumé.

[...]

### Answer: strong Sauvignon blanc is a green-skinned grape variety that originates from the city
of Bordeaux in France. The grape most likely gets its name from the French words sauvage ("wild")
and blanc ("white") due to its early origins as an indigenous grape in South West France. strong
```

# More on this strategy I have used

- Fine tune the model on a QA dataset;
- Identify heads that have a tendency to spot the rationale;
- Improve this inherent ability of the head with RL using the attention/rationale overlap as a metric;
- Access the paper:  
<https://hdl.handle.net/2268/322654>

## Exploration of Closed-Domain Question Answering Explainability Methods With a Sentence-Level Rationale Dataset

Lize Pirenne\*, Samy Mokeddem\*, Damien Ernst, Gilles Louppe

\*Equal contributions, Université de Liège,  
lize.pirenne@uliege, samy.mokeddem@uliege, dernst@uliege, g.louppe@uliege

### Abstract

In this paper, we address the problem of Rationale Extraction (RE) from Natural Language Processing: given a context ( $C$ ), a related question ( $Q$ ) and its answer ( $A$ ), the task is to find the best sentence-level rationale ( $R^*$ ). This rationale is loosely defined as being the subset of sentences of the context  $C$  such that producing  $A$  would require at least  $R^*$ . We have constructed a dataset where each entry is composed of the four terms ( $C$ ,  $Q$ ,  $A$ ,  $R^*$ ) to explore different methods in the particular case where the answer is one or multiple full sentences. The methods studied are based on TF-IDF scores, embedding similarity, classifiers and attention and have been evaluated using a sentence overlap metric akin to the Intersection over Union (IoU). Results show that the best scores were achieved by the classifier-based approach. Additionally, we observe the growing difficulty of finding  $R$  as the number of sentences in the context increased. Finally, we underlined a correlation in the case of the attention-based method between its performance and the ability of the underlying large language model to provide given  $C$  and  $Q$  an answer similar to  $A$ .

avoiding hallucinations and ensuring that the answer is grounded in reality is a priority.

Using the hypothesis that there is no redundant statement inside the context, we can identify the smallest set of sentences in the context that is required for producing the answer to the question, which we call the sentence-level rationale. For conciseness, we will refer to it as the rationale.

Extracting the rationale of an answer  $A$  from a given context  $C$  and a question  $Q$  offers significant benefits for CQA systems (Sun et al. 2022). Indeed, they enhance explainability: by identifying the rationale behind an answer, users can gain insights into the decision-making process of the underlying model of the system and assess its reliability. This is particularly valuable in domains demanding high levels of trust and transparency, such as healthcare (Ribeiro, Singh, and Guestrin 2016) or legal applications (Chalkidis et al. 2021a). Furthermore, finding the rationale can potentially improve the quality of generated responses. For example, research suggests that leveraging the rationale during prompt engineering can lead to better generation outcomes (Krishna et al. 2023). Others implicitly