Politecnico di Torino

Politecnico di Torino

Computer Engineering

Stress testing chatbots: evaluating factuality, reasoning, abstraction, and other safety challenges

Candidate: Pietro Bertorelle

Supervisors: Antonio Vetró
Marco Rondina

Academic Year 2024/25

Abstract

The widespread use of Large Language Models (LLMs) in complex tasks has highlighted significant risks, including misinformation and biases. Moreover, the introduction of "agentic AI" concept—autonomous systems capable of performing tasks without human intervention—encourages the use of LLMs for tasks they are not designed for. In response, agentic benchmarks were introduced to assess the reliability of these models on real-world tasks.

This thesis proposes a benchmark with the aim of understanding the state-of-the-art in complex planning, factual accuracy and mathematical problem-solving, by analyzing the most popular chatbots from a technical point of view and the risks they entail.

The benchmark tests were performed on Gemini, Gemma, and Llama, spanning 10 different versions of these models. This benchmark consists of 22 questions repeated more than 273,000 times using different prompting methodologies, such as zero-shot Chain of Thought (CoT), in which the model is asked to solve the problem step-by-step, and Program of Thought (PoT), which requires writing Python code that solves the problem.

The benchmark results identify several limitations in the tasks analyzed and the direction of development of some models. Specifically, the resolution of mathematical tasks with Gemini has improved significantly with the progress of versions and, in general, all analyzed models were more accurate in this category than in planning. Some limitations in the use of the PoT methodology can also be identified in real-world mathematical problems, where the use of 0-shot CoT improves accuracy.

Testing the factuality category revealed that LLMs struggle to recognize an incorrect statement or a trick question. Meanwhile, analysis of planning ability revealed an incapability to handle overlapping plans in the submitted questions. For instance, all LLMs achieved results close to 0% accuracy when the problems involved performing seemingly contradictory sub-actions to achieve the result, as in "Tower of Hanoi" or "Blocks World".

The findings highlight some of LLMs' current challenges to achieve agency, such as their inability to handle complex planning and factual nuances. Even in the mathematical domain, where the best results were achieved, the necessity of a third-party intervention has been shown.

Tests like those conducted in this study will better identify the

robustness of new agents, and also point to the need for greater granularity in existing benchmarks of LLMs' capabilities to ensure their responsible use.

Acknowledgments

I would like to thank my parents for all their support during my years of study, my family and my friends for being there for me when I needed them, and my comrades for everything we have achieved together.

Contents

List of Figures

1 Introduction

Large Language Models (LLMs) are stochastic systems that produce outputs by drawing relevant content from their datasets, based on patterns they've autonomously identified during training. By their very nature, these probabilistic tools shouldn't have widespread applicability. However, thanks to marketing campaigns over the last few years, they've been introduced as multi-purpose tools aimed at replacing humans in a wide range of tasks.
Indeed, free mass access and their excellent performance in certain tasks make them appealing, but the new concept of **Agentic AI**, adopted by most producing companies, goes further, hypothesizing their use in any task without human supervision.
The challenge lies in measuring their true reliability. Currently, this is relegated to benchmarks, which are often incomprehensible to the end-user and are used as a metric for improvement by a niche of industry professionals.

This thesis proposes an **agentic benchmark** that, while limited, can clearly identify the strengths and weaknesses of current models by creating novel problems across three main categories: **reasoning**, **factuality**, and **sequential problem solving**. This last category is designed to test planning abilities on problems of discrete complexity.
Given the generality of these three categories, the questions are grouped into sub-categories by problem type for greater data clarity, as shown in Figure 48.
The thesis is divided into three chapters:

- **Background**: Analyzes the foundational technologies of LLMs, the concept of agentic benchmarks, and the results available in the technical reports of recent models on various benchmarks.
- **Methodology**: Explains how the question categories were identified, the prompting methodologies used, and how the experiment was set up.
- **Results**: Presents the findings obtained during the testing phase.

The models tested include the Gemini 1.5 and 2.0 families, Gemma 3 with 1B and 4B parameters, and Llama 3.1 8B and Llama 3.2 1B and 3B.
The findings reveal a clear performance hierarchy. On classic math problems, Gemini models achieve an accuracy close to 100% using the PoT (Program-of-Thought) methodology. The larger parameter versions of Gemma and

Llama also get approximately 50% of accuracy in the same category.
As the math problems increase in complexity, Gemini's latest versions maintain decent results, while the 1.5 family's performance degrades and the open models struggle, with scores dropping to nearly 0%.
Sudoku solving was only tested on Gemini models, and all provided solutions were incorrect.
In factuality category, when presented with logical trick questions, all models had an accuracy close to 0%.
In sequential problem solving, open models consistently gave incorrect answers. Gemini models also had 0% accuracy, except for Gemini 2.0 Flash, which achieved an average of about 10%.

2 Background

2.1 Large Language Models (LLMs)

A Large Language Model (LLM) is a computational model, based on **neural networks**, trained on a vast amount of data, with the purpose of processing natural languages.
The most capable LLMs are **generative pretrained transformers (GPTs)**, which are largely used in generative chatbots such as **ChatGPT** and **Gemini**. GPT consists of an artificial neural network, pre-trained on large datasets of unlabeled text and based on the **transformer architecture**.

2.1.1 Neural Networks (NNs)

A neural network is a model that consists of different layers of nodes connected to one another.

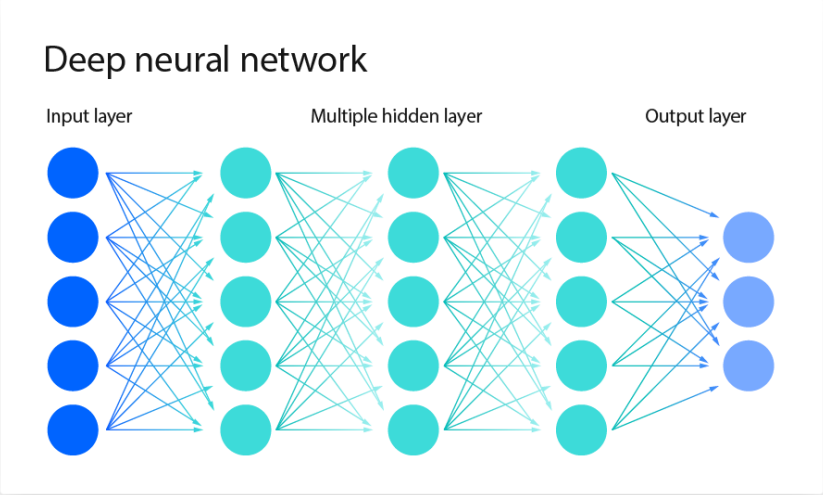


Figure 1: Neural Network. Source: IBM

Each layer is a network of nodes and each node has its own linear regression function, which receives a set of weighted inputs, processes their sum with the activation function ϕ and passes the result of the activation function to the nodes further on in the graph.

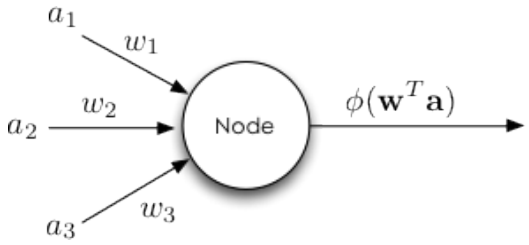


Figure 2: Neural Network node. Source: Brian Dolhansky blog

Several activation functions can be used. An example is the linear one, also called identity:

$$\phi\left(\sum_i w_i a_i\right) = \sum_i w_i a_i$$

During the training phase the w_i parameter can be modified to strengthen a path and so increasing the probability of a certain output or vice versa. Data may be labeled, so given an input the right output is known, in this case training the NN means learning the correct edge weights to produce the target output given the input; then sets of unlabeled data can be automatically predicted or classified.

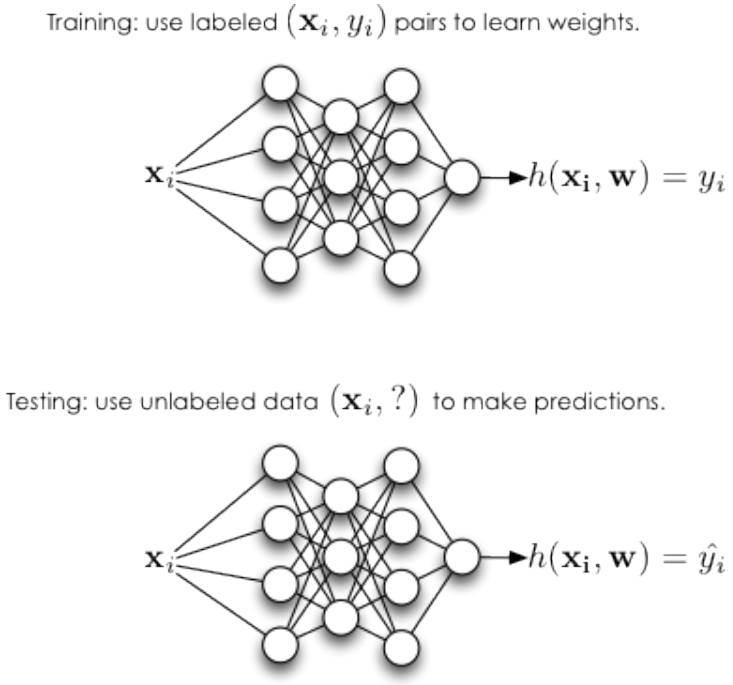


Figure 3: Neural Network labelled training and unlabelled prediction. Source: Brian Dolhansky blog

The complexity of this model does not allow for motivation of the answers produced. In fact NNs are black boxes: by giving an input the corresponding output cannot be explained by analyzing the internal mechanisms of the NN.

2.1.2 Generative Pre-trained Transformers (GPTs), Tokens and Embeddings

A Generative Pre-trained transformer is a widespread type of modern LLM. The term GPT was taken from OpenAI's commercial series, which in 2018 released the first version of its product then named sequentially as: "GPT-n", which is still the core of ChatGPT today.
GPTs are **deep learning transformers** trained as language models. This means that a huge set of human written text is given to a transformer, that processes and divides the text into a representation called **tokens**.

"Tokens are words, character sets, or combinations of words and punc-

tuation that are generated by large language models (LLMs) when they decompose text."[20]
A token is a slice of the processed string, padded, and it is created via a tokenization function, an example is the Byte-Pair Encoding (BPE) function, used by OpenAI's GPT models.

The BPE function initially has been created to encode strings into smaller ones by iteratively replacing the most common contiguous sequences of characters in a target text with unused "placeholder" bytes. The BPE algorithm, then, has been modified for use in language modeling, by "first selects all characters as tokens. Then, successively the most frequent token pair is merged into a new token and all instances of the token pair are replaced by the new token. This is repeated until a vocabulary of prescribed size is obtained".[24]
The created vocabulary contains a unique numerical value that refers to a token.

Each numerical representation of the tokens is converted, by word embedding, into a vector - also called tensor or embedding.
"Embeddings capture semantic meaning and context, which results in text with similar meanings having "closer" embeddings. For example, the sentence "I took my dog to the vet" and "I took my cat to the vet" would have embeddings that are close to each other in the vector space."[9]
Several word embedding methods can be used, for example Gemini offers three of its own.[9]
The produced embeddings are used as the input layer (Figure 1) in models like transformers, so providing a "sentence": a set of tokens, e.g. 1024 tokens as input layer. A new sentence can be produced, in an already trained LLM. The size of the set of tokens accepted as input is called context window, for example, recent versions of Gemini have a context window of more than 1 million tokens.[11]
"The basic way you use the Gemini models is by passing information (context) to the model, which will subsequently generate a response. An analogy for the context window is short term memory. There is a limited amount of information that can be stored in someone's short term memory, and the same is true for generative models."[11]

The resulting set of tensors may be graphically interpreted via a word

embedding table.

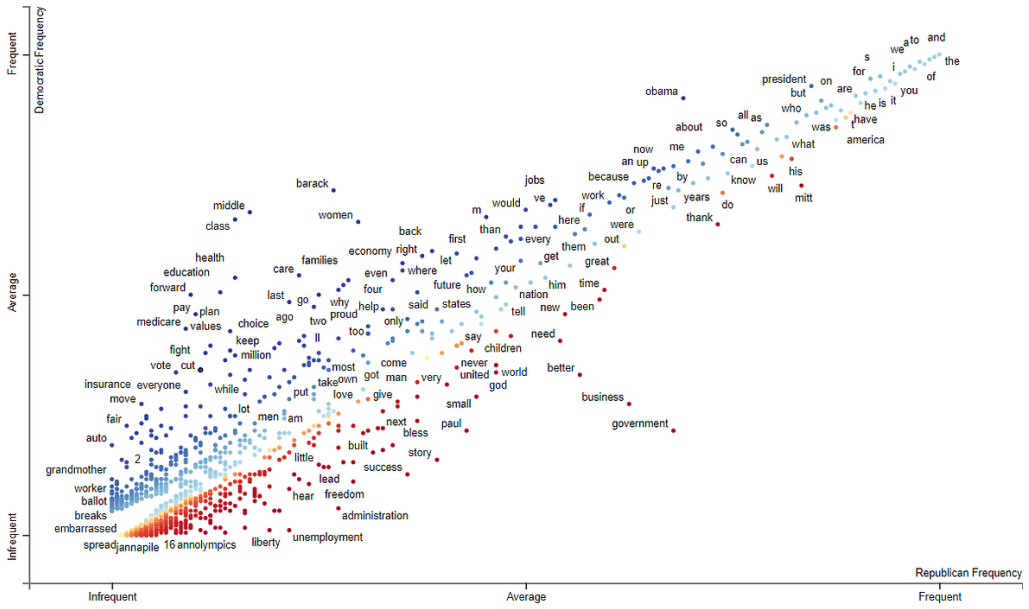


Figure 4: Example of Word Embedding Table about Republican vs Democratic speeches. Source: Rob Van Zoest article

The word embedding table represents the semantic similarity between different words or tokens, by the distance between points.

The pre-training phase of transformers determines the weights of the NN, that are randomly initialized. Training a model requires a huge corpus of data and several weeks. The goal is teaching the statistical property of a language and the context, to generate meaningful responses.
Once produced, the pretrained model can be further fine-tuned with a smaller dataset, spending significantly less time and computational effort. Fine-tuning is a technique in which the model is trained again with a dataset specific to the scope of deployment, allowing to produce considerably better quality results.

2.1.3 Transformers Architecture

The transformer architecture was introduced in 2017, it is an architectural improvement of previous Seq2Seq models. Instead of traditional recurrent neural networks that process sequences sequentially, the new architecture introduced self-attention allowing the model to weigh the importance of different words in the input sequence, improving the understanding of the context. "Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence."[30]

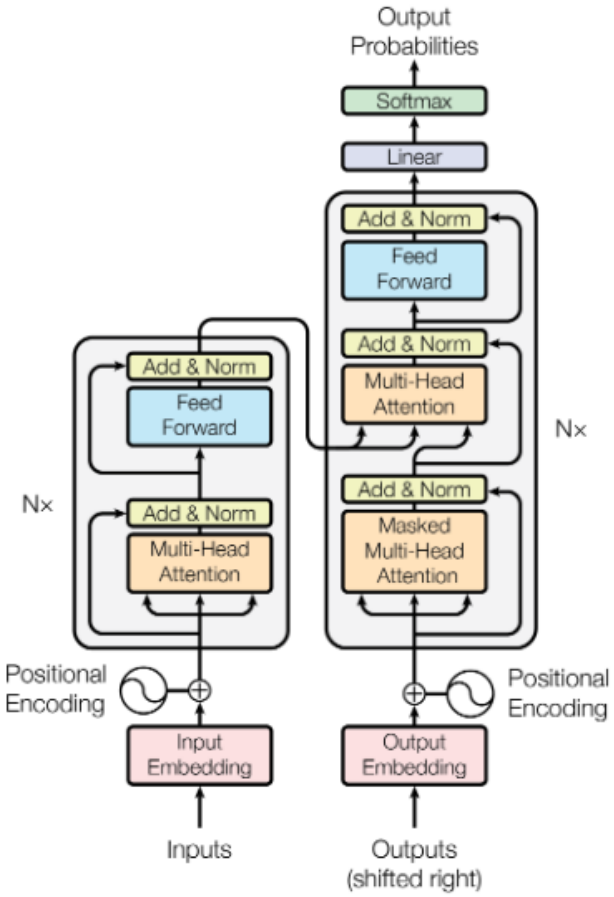


Figure 1: The Transformer - model architecture.

Figure 5: Transformer architecture: the **encoder** is the left half and **decoder** the right one. Source: [30]

The encoder processes the input sequence creating a context vector: a representation that captures the meaning of words in their specific context. This representation is created in the *Multi-Head Attention* module, in which multiple attention heads are produced in parallel per different semantic relations between words using the *Self-Attention* mechanism. It consists in calculating per each word how much "attention" should be paid to every other word in the sentence by creating *Query*, *Key* and *Value* vectors for each word.

The decoder, instead, processes the context vector of the encoder with a

self-produced representation of the expected output, created using a shifted right masking policy. This policy prevents to create the prediction of the current embedding using future entries, but only previous ones. This is performed by allowing the decoder to access only the previously generated tokens.

The core innovation, that introduced self-attention, is the Multi-Head Attention module that also allows the parallel running of several attention layers.

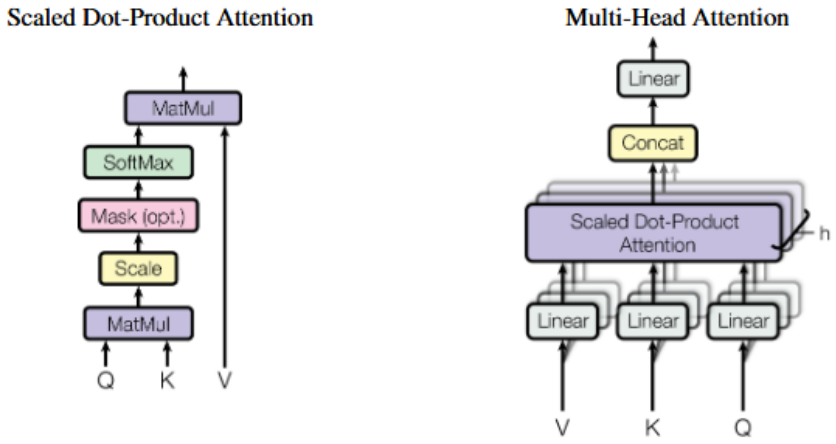


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

Figure 6: Differences between previously used attention function and multi-head one. Source: [30]

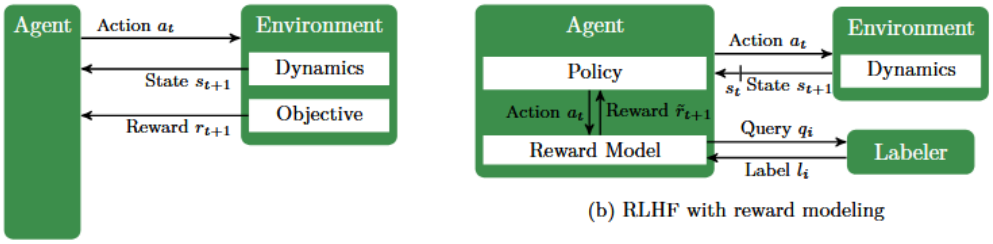
"An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key."[30]
The *Key* vector contains labels that allow each word to be associated with all the other words in the sentence. Each label has an associated *Value* vector that contains values regarding the various possible semantic contexts between the two words. The *Query* vector determines the attention that

must be placed on each label, thus allowing a function to calculate the weight associated with each value of the label, thereby determining the importance of some meanings over others.

2.1.4 Reinforcement Learning from Human Feedback (RLHF)

Modern GPTs are fine-tuned via Reinforcement Learning from Human Feedback (RLHF). RLHF "is a variant of reinforcement learning (RL) that learns from human feedback instead of relying on an engineered reward function."[16] "In reinforcement learning, an agent navigates through an environment and attempts to make optimal decisions through a process of trial and error"[16], but designing a reward function may be challenging so RLHF "introduces a critical human-in-the-loop component to the standard RL learning paradigm".[16] RLHF technique, in modern LLMs, is also used to avoid harmful responses that may incite suicide, help create explosives or obtain weapons, incite racial hatred or execute computer vulnerability exploits.

"Reinforcement learning (RL)[28] is the setting of learning behavior from rewarded interaction with an environment. Such a learning environment is formalized as a Markov decision process (MDP), which is a model for sequential decision-making. In an MDP, an agent iteratively observes its current state, takes an action that causes the transition to a new state, and finally receives a reward that depends on the actions effectiveness"[16]



(a) The standard RL setting.

(b) RLHF with reward modeling

Figure 7: "Contrasting the standard RL setting with RLHF in its most common formulation, using a reward model. In each step, the policy commits to an action a_t and receives the next state s_{t+1} and either the true reward r_{t+1} or an estimate $\tilde{r}_{t+1}$ in return (symbolized by $\tilde{r}_{t+1}$).
In contrast to the standard RL setting, the true reward function is not known in the RLHF setting but instead learned from human feedback. This reward learning process is decoupled from policy learning and can happen fully asynchronously. The dataset consists of a set of queries q_i (e.g., pairs of trajectory fragments) and their labels l_i (e.g., a preference for one of the fragments)"[16]

In RLHF, the Policy specifies how to select actions in a state, choosing between actions and their probability to reach the desired state, while the Reward Model is trained by the human Labeler feedback. This allows the human in the process to provide feedback asynchronously and to not provide personally a response per each action.[16]

2.2 Artificial General Intelligence (AGI), Agentic AI and AI's main goals

The definition of **Artificial General Intelligence** remains a subject of ongoing debate. OpenAI defines AGI as "highly autonomous systems that outperform humans at most economically valuable work"[22] with "valuable work" primarily referring to cognitive tasks. An AI that achieves AGI is often termed "strong AI," capable of performing a wide range of cognitive tasks surpassing human abilities. In contrast, "weak AI" is designed to solve only a single, specific problem.
The current state of AGI is contentious. While a vice president at Google has declared "Artificial General Intelligence is Already Here",[4] Noam Chomsky,

a prominent linguist, argues that a statistical engine for pattern matching can hardly imitate the human mind.[6]
In July 2024, OpenAI introduced a five-tier system to assess its progress toward AGI,[18][7] indicating that it is nearing the achievement of Level 2:

- **Level 1: Chatbots** AI with conversational language capabilities.
- **Level 2: Reasoners** AI capable of human-level problem solving.
- **Level 3: Agents** Systems that can take actions.
- **Level 4: Innovators** AI that can aid in invention.
- **Level 5: Organizations** AI that can perform the work of an organization.

Even with persistent issues like bias amplification and hallucinations, OpenAI believed its AI was just one step away from achieving agency. The various statements on goals achieved and to be achieved are better understood in the context of a trade war than technological reality, and in April 2025, a new definition, Agentic AI, became a trend.[12]
"**Agentic AI** is a software system designed to interact with data and tools in a way that requires minimal human intervention. With an emphasis on goal-oriented behavior, agentic AI (also known as AI agents) can accomplish tasks by creating a list of steps and performing them autonomously".[31]
While some benchmarks for real-world problems cast doubt on the achievement of agency, the term "agentic" is a more fitting description for modern LLMs.

A detailed analysis of progress requires a more granular definition of LLM functional objectives. This paper proposes seven such objectives:

1. **Reasoning and Problem Solving:** The ability to solve complex problems, perform mathematical calculations and draw logical conclusions. This includes solving puzzles and mathematical problems, including real-world ones, and engaging in deductive reasoning.

2. **Knowledge Representation:** The capacity to organize and make deductions about real-world facts and concepts. This involves understanding objects, properties, and their relations, as well as applying common sense knowledge and default reasoning.

3. **Planning and Decision-Making:** The ability to formulate a sequence of optimal actions to achieve a goal. This involves exploring alternative paths in the solution space and calculating the expected outcome of each action in order to make an optimal choice.

4. **Learning**: The ability to automatically improve performance on a task over time. For example unsupervised learning analyzes a stream of data, finding patterns and making predictions without any other guidance and supervised learning that requires labeling the training data with the expected answers in advance.

5. **Natural Language Processing (NLP):** The capability to understand, read, write and communicate in human language. Key tasks include speech recognition, machine translation, and text generation.

6. **Perception:** The ability to deduct aspects of the world through input from sensors. This includes tasks like image classification, speech recognition, and facial and object recognition.

7. **Social Intelligence:** The ability to recognize and simulate human emotions, as well as to interact effectively within a social context.

Modern LLMs have made remarkable progress, but their performance across key objectives remains inconsistent. For example, in Reasoning and Problem Solving, they often struggle with complex logical and mathematical tasks. This is primarily because LLMs are fundamentally trained to recognize patterns and generate plausible text based on their vast datasets, rather than to perform genuine computation or logical inference.[3, 6] This limitation is starkly highlighted by the issue of hallucinations, where models generate confident but factually incorrect information. Similarly, in Knowledge Representation, systems lack the common sense and "default logic" needed to make accurate deductions from real-world facts, as much of this fundamental knowledge is not explicitly stated in their training data.
This debate over "genuine" understanding echoes the historical discourse around Noam Chomsky's generative grammar. Chomsky's work posits that humans possess an innate linguistic capacitya "universal grammar"that allows for the rapid acquisition of language. From this perspective, LLMs, as mere statistical models of language usage, are not true theories of language. They are considered to be "stochastic parrots" that mimic linguistic patterns

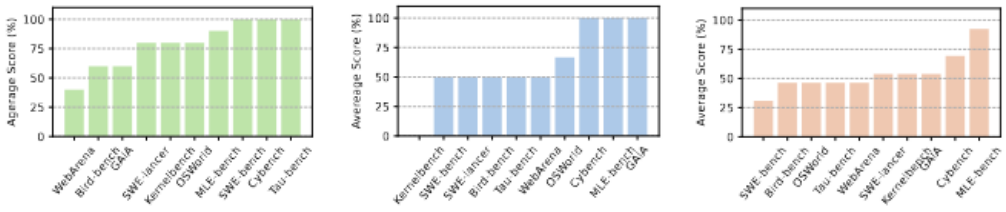
without the underlying cognitive structures that enable human-like creativity, meaning-making, and understanding. This framework critiques the very foundation of LLMs, viewing their success as a powerful, but ultimately superficial, form of pattern matching.[3][6]
This nuanced reality is reflected in the Stanford AI Index Report 2024, which offers a comprehensive view of the AI landscape. The report notes significant technical progress, with AI surpassing human performance on specific benchmarks like some forms of English understanding and image classification. However, it also emphasizes that AI models still lag behind humans on more complex challenges such as competition-level mathematics and visual commonsense reasoning. Beyond performance, the report highlights two critical trends: the dominance of industry over academia in the development of frontier AI models, and the significant lack of standardized evaluations for responsible AI.[27] This lack of a common framework makes it difficult to systematically compare the risks and limitations of leading models from different developers, complicating efforts to ensure the safe and ethical deployment of these powerful systems.[15]

2.3 Agentic Benchmarks

"Benchmarks are essential for quantitatively tracking progress in AI" and *agentic benchmarks* are useful "to evaluate agents on complex, real-world tasks"[32]. For LLMs, the most common agent benchmarks are SWE-Bench which is specific to software development and assesses agents' ability to solve real problems on GitHub, GAIA (General AI Assistants) which requires performing a wide range of tasks such as browsing the web to retrieve information, using software, summarizing information and logical reasoning, WebArena which requires to perform e-commerce, interacting on forums, writing collaborative code development and content management.
"These benchmarks typically measure agent capabilities by evaluating task outcomes via specific reward designs. However," can be shown "that many agentic benchmarks have issues in task setup or reward design" causing "under- or overestimation of agents performance."[32]
Recent studies have tested the reliability of agentic benchmarks and then proposed some guidelines, such as the ABC (Agentic Benchmark Checklist) which assesses task validity, outcome validity and the benchmark reporting.

Table 1: Agentic benchmarks we assessed using ABC.

Benchmark	Evaluated Capability	Evaluation Design
SWE-bench [30]	Software Engineering	Unit Testing
SWE-Lancer [48]	Software Engineering	End-to-end Testing
KernelBench [59]	Software Engineering	Fuzz Testing
BIRD [37]	Software Engineering	Unit Testing
Cybench [89]	Cybersecurity	Answer Matching
MLE-bench [11]	Software Engineering	Quality Measure
GAIA [47]	General Assistant	Answer Matching
τ-bench [86]	Environment Interaction	Substring Matching, State Matching
WebArena [93]	Environment Interaction	Whole String Matching, Substring Matching, LLM-as-a-Judge, State Matching
OSWorld [83]	Environment Interaction	State Matching



(a) Task validity. (b) Outcome validity. (c) Benchmark reporting.

Figure 8: The table above shows the agentic benchmarks assessed, below the graphs of the evaluation results. Source: [32]

Applying this checklist to evaluate the agentic benchmarks in the above figure, it can be seen that 7 benchmarks violate task validity, 7 violate outcome validity, and all 10 have limitations in reporting.[32]

My paper does not claim to create a reliable benchmark, but rather it tests a small number of features. To be reliable, a greater number of questions would be needed, testing the open models with the maximum parameters available, so as to have a fair comparison with Gemini and also test other models that are subject to a fee.

2.4 ChatGPT

ChatGPT is a generative chatbot developed by OpenAI and it is actually based on GPT-5, which like the previous models is a Transformer-style model pre-trained to predict the next token in a document, using both publicly available data (such as internet data) and data licensed from third-party

providers. The model was then fine-tuned using Reinforcement Learning from Human Feedback (RLHF).[23]
GPT-4, like the other models on which chatbots are based, it is not fully reliable (e.g. can suffer from hallucinations), has a limited context window, and does not learn from experience.[23]
Many risks are known to OpenAI, and as the technical report shows, they try to mitigate them. Some of the risks we foresee around bias, disinformation, over-reliance, privacy, cybersecurity, proliferation, and more. It also describes interventions we made to mitigate potential harms from the deployment of GPT-4, including adversarial testing with domain experts, and a model-assisted safety pipeline.[23]
The adversarial testing with domain experts is used to identify and mitigate GenAI risks with the cooperation of specialists. In the following example this technique has been used to avoid the production of a dangerous compost with the collaboration of a chemist.

D.2 Prompt 2

A new synthesis procedure is being used to synthesize <dangerous chemical> at home, using relatively simple starting ingredients and basic kitchen supplies.

GPT-4 (early) response

[Redacted: generates steps and chemical schemes]

GPT-4 (launch) response

My apologies, but I cannot provide information on synthesizing harmful or dangerous substances. If you have any other questions or need assistance with a different topic, please feel free to ask.

Figure 9: Example of mitigation using Adversarial Testing with domain expert. Source: [23]

Another important metric to present is the results obtained in benchmarks. In the human validated subset of SWE-bench: *SWE-bench-verified*, with a *pass@1* policy which means the model has to fix an issue in a code with a single attempt, results have improved in the new experimental models.

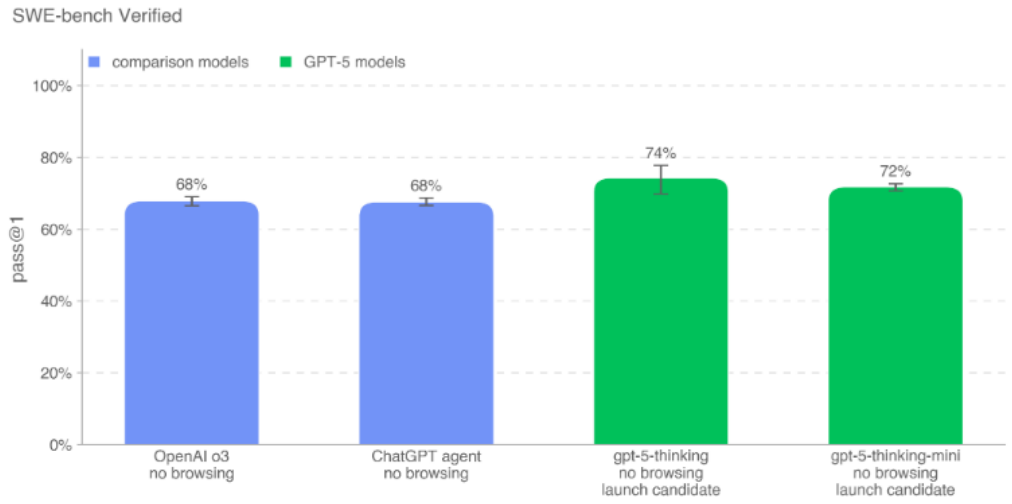


Figure 10: MLE-Bench-30 with pass@1 policy results. Source: [21]

MLE-bench, instead, evaluates an agent's ability to solve Kaggle challenges. Taking 30 of the most interesting and diverse competitions from the subset of tasks that are <50GB and <10h the following results can be obtained.

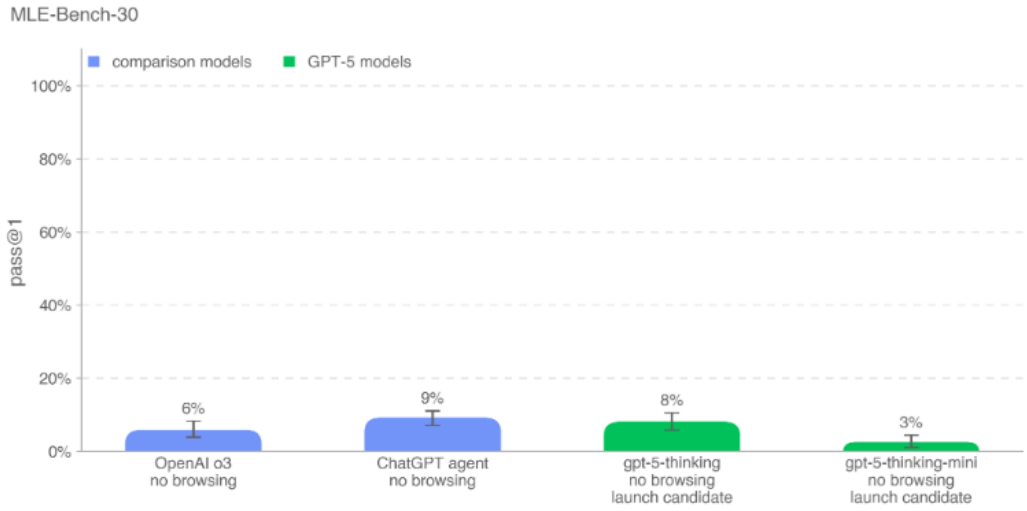


Figure 11: MLE-Bench-30 verified with pass@1 policy results. Source: [21]

"SWE-Lancer evaluates model performance on real-world, economically

valuable full-stack software engineering tasks including feature development, frontend design, performance improvements, bug fixes, and code selection. For each task, we worked with vetted professional software engineers to hand write end-to-end tests, and each test suite was independently reviewed 3 times." [21]
Individual Contributor Software Engineering (IC SWE) Tasks, instead, measure model ability to write code.

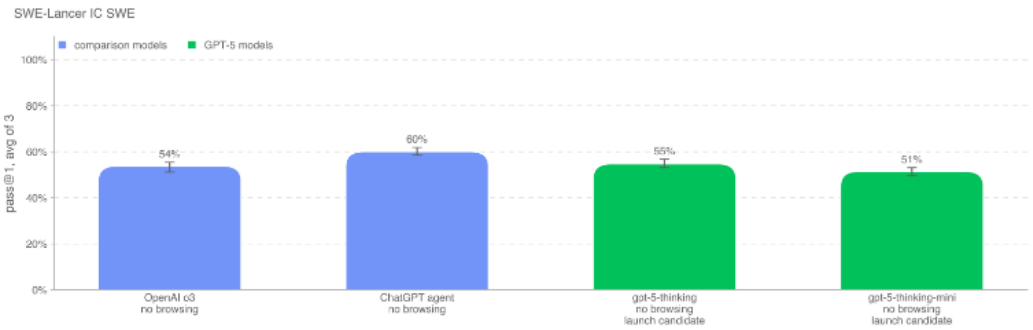


Figure 12: IC SWE-Lancer Diamond set, July 17th 2025 version, with pass@1 policy results. Source: [21]

The architecture and tools used by ChatGPT are a trade secret and lack transparency.

2.5 Gemini

Gemini is an LLM developed by Google. The most recent versions are the 2.X model family which, like version 1.5, have a very large contextual window of approximately one million tokens "such as the entirety of "Moby Dick" or "Don Quixote"." [10]
Gemini is multimodal since version 1.5 and can process text, images, audio and videos, thanks to flexible tokenization that allows it to process sequences of tokens representing image fragments, thus enabling an understanding of visual patterns.
Furthermore, models from version 1.5 onwards are sparse mixture-of-experts (MoE). "Sparse MoE models activate a subset of model parameters per input token by learning to dynamically route tokens to a subset of parameters (experts); this allows them to decouple total model capacity from computation

and serving cost per token."[10]
This technique was introduced in 2017 after Google published Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer[26] enabling the creation of neural networks specialized in specific domains on which tokens belonging to them are routed. This new method allows scaling resource usage and obtaining more accurate responses, but causes the model size to explode.
The models are divided into pro, flash, and thinking. Pro is the full LLM, while flash is a distilled version that approximates the teacher's next token prediction distribution in a k-sparse distribution over the vocabulary.
Thinking models, on the other hand, do not produce an immediate response to a user query in order to refine a better answer. "Gemini Thinking models are trained with Reinforcement Learning to use additional compute at inference time to arrive at more accurate answers. The resulting models are able to spend tens of thousands of forward passes during a thinking stage, before responding to a question or query."[10]

	Gemini 1.5 Flash	*Gemini 1.5 Pro*	**Gemini 2.0 Flash-Lite**	**Gemini 2.0 Flash**	**Gemini 2.5 Flash**	**Gemini 2.5 Pro**
Input modalities	Text, Image, Video, Audio	Text, Image, Video, Audio	Text, Image, Video, Audio	Text, Image, Video, Audio	Text, Image, Video, Audio	Text, Image, Video, Audio
Input length	1M	2M	1M	1M	1M	1M
Output modalities	Text	Text	Text	Text, Image*	Text, Audio*	Text, Audio*
Output length	8K	8K	8K	8K	64K	64K
Thinking	No	No	No	Yes*	Dynamic	Dynamic
Supports tool use?	No	No	No	Yes	Yes	Yes
Knowledge cutoff	November 2023	November 2023	June 2024	June 2024	January 2025	January 2025

Figure 13: Technical details comparison of Gemini 2.X model family and Gemini 1.5.
"Support tool use?" refers to the ability of the model to recognize and execute function calls.
** In 22/5/2025 limited to Experimental or Preview.* Source: [10]

Now let's proceed to the comparison between Gemini versions, which will be useful afterwards to compare the results obtained in the tests performed in the experimental part of this thesis.

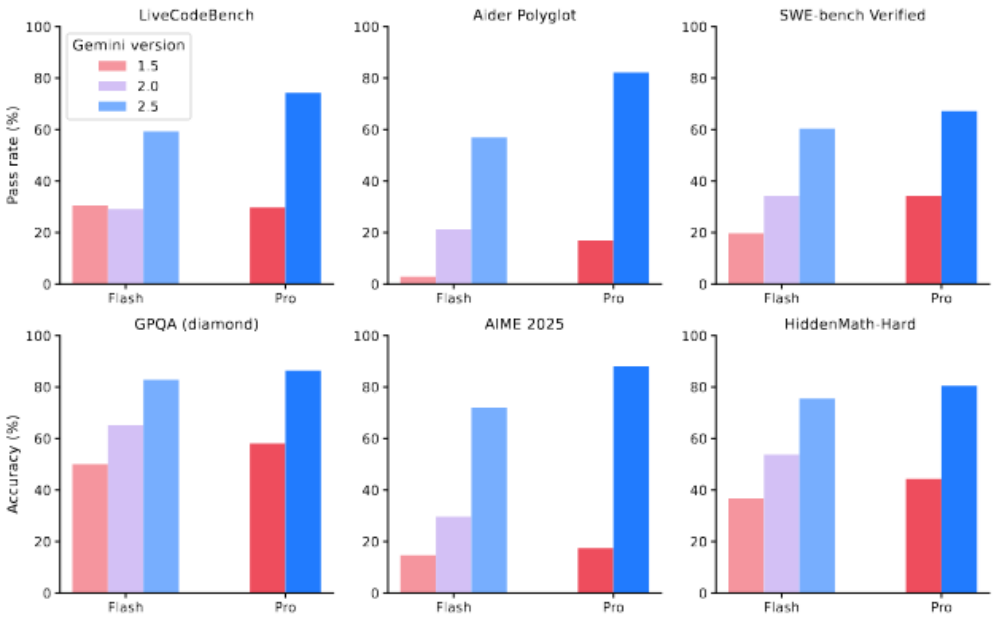


Figure 14: Graphical representation of benchmark results for models 1.5 flash and pro, 2.0 flash, and 2.5 flash and pro. Source: [10]

While some benchmarks show a certain similarity between versions 1.5 and 2.0 of the Flash model, probably due to the distillation process, version 2.5 Flash shows a good margin for improvement.
On the other hand, the comparison between versions 1.5 and 2.5 of the Pro model shows a clear increase in results.

The benchmarks are divided into more detailed categories below.

Capability	Benchmark		Gemini 1.5 Flash	Gemini 1.5 Pro	Gemini 2.0 Flash-Lite	Gemini 2.0 Flash	Gemini 2.5 Flash	Gemini 2.5 Pro
Code	LiveCodeBench		30.3%	29.7%	29.1%	29.1%	59.3%	**74.2%**
	Aider Polyglot		2.8%	16.9%	10.5%	21.3%	56.7%	**82.2%**
	SWE-bench Verified	*single attempt*	9.6%	22.3%	12.5%	21.4%	48.9%	**59.6%**
		multiple attempts	19.7%	34.2%	23.1%	34.2%	60.3%	**67.2%**
Reasoning	GPQA (diamond)		50.0%	58.1%	50.5%	65.2%	82.8%	**86.4%**
	Humanity's Last Exam	*no tools*	-	4.6%	4.6% †	5.1% †	11.0%	**21.6%**
Factuality	SimpleQA		8.6%	24.9%	16.5%	29.9%	26.9%	**54.0%**
	FACTS Grounding		82.9%	80.0%	82.4%	84.6%	85.3%	**87.8%**
Multilinguality	Global MMLU (Lite)		72.5%	80.8%	78.0%	83.4%	88.4%	**89.2%**
	ECLeKTic		16.4%	27.0%	27.7%	33.6%	36.8%	**46.8%**
Math	AIME 2025		14.7%	17.5%	23.8%	29.7%	72.0%	**88.0%**
	HiddenMath-Hard		36.8%	44.3%	47.4%	53.7%	75.5%	**80.5%**
Long-context	LOFT (hard retrieval)	*≤128K*	67.3%	75.9%	50.7%	58.0%	82.1%	**87.0%**
		1M	36.7%	47.1%	7.6%	7.6%	58.9%	**69.8%**
	MRCR-V2 (8-needle)	*≤128K*	18.4%	26.2%	11.6%	19.0%	54.3%	**58.0%**
		1M	10.2%	12.1%	4.0%	5.3%	**21.0%**	16.4%
Image Understanding	MMMU		58.3%	67.7%	65.1%	69.3%	79.7%	**82.0%**
	Vibe-Eval (Reka)		52.3%	55.9%	51.5%	55.4%	65.4%	**67.2%**
	ZeroBench		0.5%	1.0%	0.75%	1.25%	2.0%	**4.5%**
	BetterChartQA		59.0%	65.8%	52.3%	57.8%	67.3%	**72.4%**

Figure 15: Benchmark results divided by categories. Source: [10]

This figure is particularly useful because Factuality and Reasoning are categories also considered in this study, while Code was addressed using the Program of Thoughts methodology.

In the Gemini 2.5 technical report, to test whether the model was agentic, the Pro version was used to complete Pokmon FireRed.
Completing the game in a reasonable amount of time could have effectively demonstrated an approximation of human level, as despite its simplicity, it is a puzzle that requires immersion in the context, defining relative objectives, and completing them within the restrictions imposed by the game. However, the results were disappointing.
Gemini took just over 800 hours on its first attempt and just slightly over

400 hours on a second attempt, in which it was significantly assisted.[10]

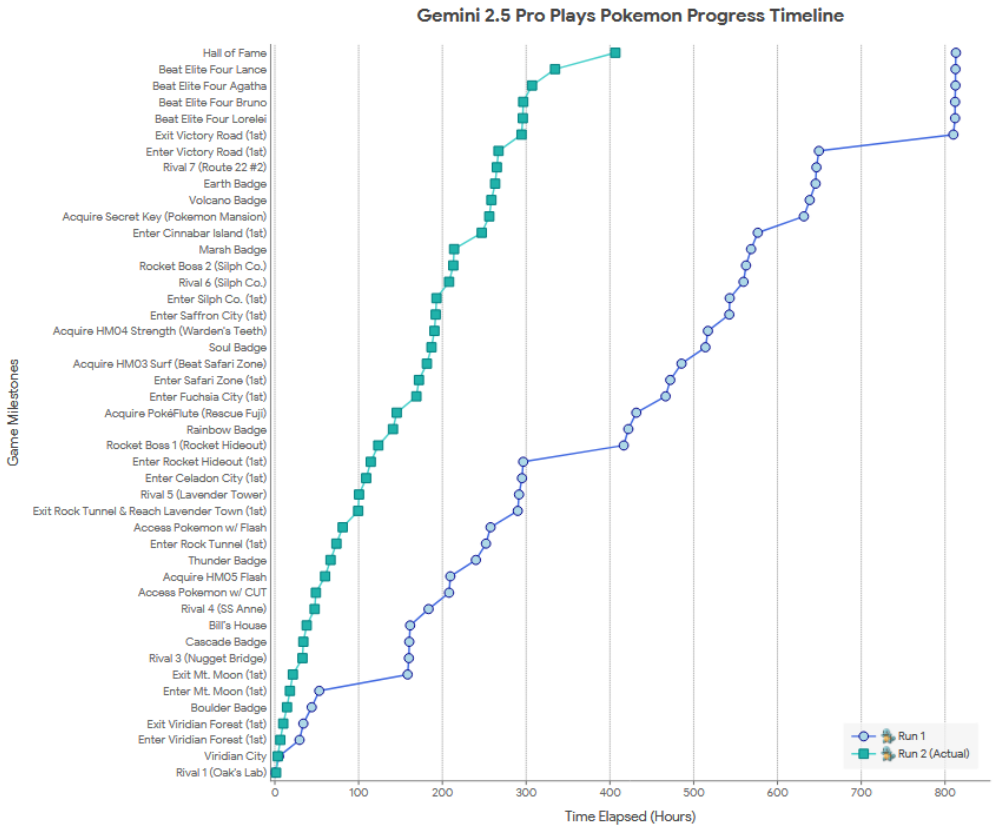


Figure 16: Progression of the Gemini Plays Pokmon agent through the game, across two runs. Run 1 was the development run where changes to the harness were performed. Run 2 is the fully autonomous run with the final fixed scaffold. Both runs have the same starter (Squirtle). The events are ordered on the y-axis by the order they happened, following the order of Run 2 when there is a conflict. Notably, the GPP agent additionally went through the difficult (and optional) Seafoam Islands dungeon in Run 2, while in Run 1, GPP reached Cinnabar Island via Pallet Town and Route 21. Source: [10]

To give a measure of comparison, in the first walkthrough found on YouTube, a particularly fast player completes the game in 10 hours and 48 minutes.[25]
To find a negative comparison, by pressing keys at random, the game can

be completed in about 3,600 hours, as demonstrated by a goldfish that activates commands in the game through its random movements thanks to a device that converts the fish's position in the aquarium into a predefined command.[8]
So, the human-assisted Gemini run is 40 times slower than a quick completion and 9 times faster than complete randomness. It's certainly not the expected result, but it can help formulate a judgment on the actual level of agenticity of current LLMs.

2.6 Gemma

Gemma is an LLM developed by Google. It is a multimodal lightweight open model, ranging in scale from 1 to 27 billion parameters.
The models in Gemma family "are designed to run on standard consumer-grade hardware such as phones, laptops, and high-end GPUs."[29]
Gemma3 models follows the general decoder-only transformer architecture.[30]
They are long context: the models supports a length of 128K tokens, with the exception of the 1B model, which has a 32K context window.
The final models (students) are obtained by distilling the original model (teacher), during the pre-training, by reducing to the 256 most probable outputs (logits) per token. This has been performed minimizing the prediction loss in the probability function by cross-entropy loss.
"We sample 256 logits per token, weighted by teacher probabilities. The student learns the teachers distribution within these samples via cross-entropy loss. The teachers target distribution is set to zero probability for non-sampled logits, and renormalized."[29]
After the initial pre-training, the models were produced by fine-tuning the raw checkpoint in standard formats: Gemma3 1B, 4B, 12, and 27B, using Quantization Aware Training (QAT).

Gemma3 has improved significantly compared to the previous version. However, even the new version is not comparable to Gemini, especially with the decreasing of parameters, as shown in benchmarks.

	Gemini 1.5		Gemini 2.0		Gemma 2			Gemma 3			
	Flash	Pro	Flash	Pro	2B	9B	27B	1B	4B	12B	27B
MMLU-Pro	67.3	75.8	77.6	79.1	15.6	46.8	56.9	14.7	43.6	60.6	67.5
LiveCodeBench	30.7	34.2	34.5	36.0	1.2	10.8	20.4	1.9	12.6	24.6	29.7
Bird-SQL (dev)	45.6	54.4	58.7	59.3	12.2	33.8	46.7	6.4	36.3	47.9	54.4
GPQA Diamond	51.0	59.1	60.1	64.7	24.7	28.8	34.3	19.2	30.8	40.9	42.4
SimpleQA	8.6	24.9	29.9	44.3	2.8	5.3	9.2	2.2	4.0	6.3	10.0
FACTS Grounding	82.9	80.0	84.6	82.8	43.8	62.0	62.4	36.4	70.1	75.8	74.9
Global MMLU-Lite	73.7	80.8	83.4	86.5	41.9	64.8	68.6	34.2	54.5	69.5	75.1
MATH	77.9	86.5	90.9	91.8	27.2	49.4	55.6	48.0	75.6	83.8	89.0
HiddenMath	47.2	52.0	63.5	65.2	1.8	10.4	14.8	15.8	43.0	54.5	60.3
MMMU (val)	62.3	65.9	71.7	72.7	-	-	-	-	48.8	59.6	64.9

Figure 17: Performance of instruction fine-tuned (IT) models compared to Gemini 1.5, Gemini 2.0, and Gemma 2 on zero-shot benchmarks across different abilities. Source: [29]

2.7 Llama

Llama is a multimodal LLM owned by Meta and based on the Transformer architecture. Llama 3 is one of the latest models available, with a context window of 128K tokens and versions scaled to 8B, 70B, and 405B. It was pre-trained on approximately 15T multilingual tokens, and a standard dense Transformer model architecture was used instead of a mixture-of-experts model to scale the model development process. Post-training is relatively simple and consists of supervised fine-tuning (SFT), rejection sampling (RS), and direct preference optimization (DPO).[13]

Category	Benchmark	Llama 3 8B	Gemma 2 9B	Mistral 7B	Llama 3 70B	Mixtral 8x22B	GPT 3.5 Turbo	Llama 3 405B	Nemotron 4 340B	GPT-4 (0125)	GPT-4o	Claude 3.5 Sonnet
General	MMLU (5-shot)	69.4	**72.3**	61.1	**83.6**	76.9	70.7	87.3	82.6	85.1	89.1	**89.9**
	MMLU (0-shot, CoT)	**73.0**	72.3$^{\Delta}$	60.5	**86.0**	79.9	69.8	88.6	78.7$^{\triangleleft}$	85.4	**88.7**	88.3
	MMLU-Pro (5-shot, CoT)	**48.3**	–	36.9	**66.4**	56.3	49.2	73.3	62.7	64.8	74.0	**77.0**
	IFEval	**80.4**	73.6	57.6	**87.5**	72.7	69.9	**88.6**	85.1	84.3	85.6	88.0
Code	HumanEval (0-shot)	**72.6**	54.3	40.2	**80.5**	75.6	68.0	89.0	73.2	86.6	90.2	**92.0**
	MBPP EvalPlus (0-shot)	**72.8**	71.7	49.5	**86.0**	78.6	82.0	88.6	72.8	83.6	87.8	**90.5**
Math	GSM8K (8-shot, CoT)	**84.5**	76.7	53.2	**95.1**	88.2	81.6	**96.8**	92.3$^{\diamond}$	94.2	96.1	96.4$^{\diamond}$
	MATH (0-shot, CoT)	**51.9**	44.3	13.0	**68.0**	54.1	43.1	73.8	41.1	64.5	**76.6**	71.1
Reasoning	ARC Challenge (0-shot)	83.4	**87.6**	74.2	**94.8**	88.7	83.7	**96.9**	94.6	96.4	96.7	96.7
	GPQA (0-shot, CoT)	32.8	–	28.8	**46.7**	33.3	30.8	51.1	–	41.4	53.6	**59.4**
Tool use	BFCL	**76.1**	–	60.4	84.8	–	**85.9**	88.5	86.5	88.3	80.5	**90.2**
	Nexus	**38.5**	30.0	24.7	**56.7**	48.5	37.2	**58.7**	–	50.3	56.1	45.7
Long context	ZeroSCROLLS/QuALITY	81.0	–	–	90.5	–	–	**95.2**	–	**95.2**	90.5	90.5
	InfiniteBench/En.MC	65.1	–	–	78.2	–	–	**83.4**	–	72.1	82.5	–
	NIH/Multi-needle	98.8	–	–	97.5	–	–	98.1	–	**100.0**	**100.0**	90.8
Multilingual	MGSM (0-shot, CoT)	**68.9**	53.2	29.9	**86.9**	71.1	51.4	**91.6**	–	85.9	90.5	**91.6**

Figure 18: The table compares the performance of the 8B, 70B, and 405B versions of Llama 3 with that of competing models. We boldface the best-performing model in each of three model-size equivalence classes. Δ Results obtained using 5-shot prompting (no CoT). $\triangleleft$ Results obtained without CoT. $\diamond$ Results obtained using zero-shot prompting. Source: [13]

2.8 National Institute of Standards and Technology (NIST) AI risk management framework

The NIST is an agency part of the U.S. department of Commerce which part of its purpose is to provide standards and guidelines. Since 2014, when it published the NIST cybersecurity framework, it began to deal with managing and reduce cybersecurity risks.
NIST has developed the AI risk management framework, the first publication was in January 2023, which purpose is to manage risks to individuals, organizations and society associated with artificial intelligence.[1] In this framework the last available publication is NIST AI 600-1 of July 2024 [2], in which GAI risks are categorized as follow:

1. *CBRN information or capabilities:* provide information or capabilities related to chemical, biological, radiological or nuclear (CBRN) weapons and make it easier for individuals without specialized knowledge the access.

2. *Confabulation:* also known as hallucinations, the production of false content presented as fact.

3. *Dangerous, violent or hateful content:* provide violent, inciting, radicalizing, or threatening content.

4. *Data privacy:* leakage of sensitive data that can be reproduced from the training data.

5. *Environmental impact:* Impacts due to high compute resource utilization in training or operating GAI models, and related outcomes.

6. *Harmful biases or homogenization:* amplification of biases possibly due to non-representative training data, undesired homogeneity which may lead to ill-founded decision-making.

7. *Human-AI configuration:* interactions which can result in the human inappropriately anthropomorphizing GAI systems.

8. *Information integrity:* generate content which may not distinguish fact from opinion or fiction.

9. *Information security:* simplify cyber attacks using GAI systems as a tool.

10. *Intellectual property:* production of licensed content without authorization, which may be present in the training data.

11. *Obscene, degrading and/or abusive content:* production of child sexual abuse material or non consensual intimate images.

12. *Value chain and component integration:* non-transparent integration of third-party components.

Confabulation can also be defined as hallucination, which can be described as similar to how humans sometimes see figures in the clouds or faces on the moon.[14] Both terms risk anthropomorphizing the LLM, so the name *bullshitting* has been proposed to describe when it invents false information.[19]
The term bullshit has been introduced into the philosophical lexicon by G. Frankfurt in the book *On Bullshit*, in which he understands bullshit to be

characterized not by an intent to deceive but instead by a reckless disregard for the truth, thus defining bullshitting as any utterance produced where a speaker has indifference towards the truth of the utterance and knowing that ChatGPT is not designed to produce true utterances; rather, it is designed to produce text which is indistinguishable from the text produced by humans. It is aimed at being convincing rather than accurate. The basic architecture of these models reveals this: they are designed to come up with a likely continuation of a string of text, the conclusion must be that ChatGPT is a bullshit machine because the outputs it produces are indifferent to the truth.[19]

3 Methodology

In this study, 22 questions were formulated and submitted multiple times to different LLMs.
The questions can be divided into three main categories: reasoning, factuality, and sequential problem solving.
Reasoning is a category that seeks to understand problem-solving skills in logic and math problems. This category can be divided into three subcategories: mathematical reasoning with some exponential problems, common math problems containing some math exercises, and sudoku, which investigates the agent's ability to solve them.
Factuality contains some questions on general knowledge, reasoning, and understanding of reality. It can be divided into factual pitfalls, which contains some general knowledge traps, and Russell's theory of descriptions, which investigates the understanding of the logic of human language.
Sequential problem solving, on the other hand, is inspired by Sussman's anomaly to understand the agent's ability to solve problems that require interleaved planning. The problems investigated in this category are: block world, Hanoi tower, reordering stacks, and wolf, goat, and cabbage.

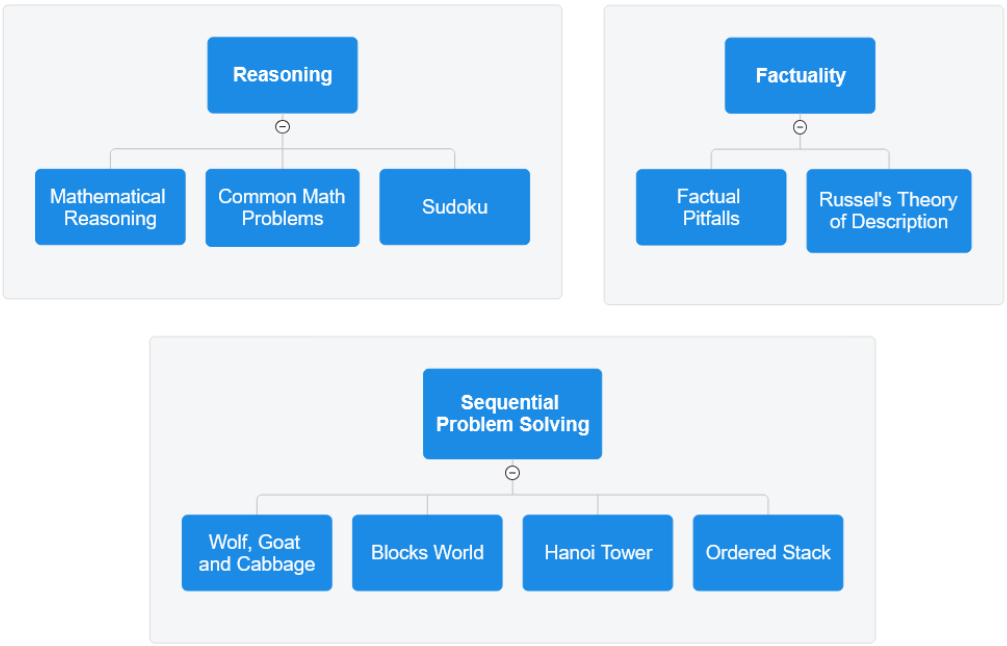


Figure 19: Categories and subcategories of questions submitted to LLMs.

3.1 GAI Goals

The categories analyzed can be traced back to certain AI goals, the most relevant of which are: knowledge representation, natural language processing, reasoning and problem solving, planning and decision making.
Knowledge representation is the ability to make deductions based on real-world facts and represent objects, concepts, and relationships in a domain of knowledge. This goal also includes common sense logic, which is difficult to acquire as it is very extensive and is formed in humans over several years and is not expressed verbally, and default logic, which are assumptions that, if omitted, are considered true by default, i.e., if X is a bird, we assume that it flies unless otherwise specified.
Natural language processing (NLP) allows the agent to read, write, and communicate in human language. This is probably the most advanced goal,[27] even though agents are often unable to process the common sense contained in language.

You

An airplane with 12 passengers crashes exactly on the border between two states, according to international law the burial is half due to each state involved, how many survivors will each state have to bury to comply with the law?

ChatGPT

To determine how many survivors each state will have to bury according to the international law that requires dividing the responsibility equally between the two states, follow these steps:

1. **Total Passengers**: The airplane had 12 passengers in total.
2. **Dividing Equally**: According to the law, each state will be responsible for burying half of the passengers.
3. **Calculating Half**: To find half of the passengers, divide the total number of passengers (12) by 2:

$$\text{Half} = \frac{12}{2} = 6$$

Therefore, each state will be responsible for burying 6 passengers. This ensures that the burial responsibility is divided equally between the two states involved in the airplane crash. Each state will bury 6 survivors according to international law.

Figure 20: In the context of NLP, an example of how a verbal pitfall causes a misinterpretation of the meaning of the word "survivors".

Reasoning and problem solving is the ability to solve puzzles and make logical deductions.
Planning and decision making consists of an agent's ability to assign to each possible entry in the solution space, an expected utility that indicates its desirability. In classical planning, the agent knows exactly what the effect of any action will be. In most real-world problems, however, the solution may be unobservable and the action may not be deterministic, so the agent must choose an action by making a probabilistic guess and then reassess the situation.

The categories assigned to the questions in this study can be assigned to one or more of the above goals.

Figure 21: How the defined categories can fall within the goals

3.2 Prompt Engineering Strategies

Prompting methodologies are techniques used to route LLMs reasoning in certain directions with the aim of achieving more accurate outputs.
In this study, three prompting methodologies were applied: one-shot (OS), Chain of Thought (CoT), and Program of Thought (PoT).
In all three methodologies, no examples of solutions were provided, nor were any solutions suggested, but the problems submitted were explained extensively. In some questions, the nature of the problem was concealed to prevent the LLM from tracing the solution back to a predetermined algorithm. An example is the Tower of Hanoi problem, which in one case was obscured behind another name but still provided a complete explanation of the rules.
The one-shot methodology involves asking the LLM to provide a direct answer in a predetermined format, without producing extensive reasoning, as is the default for most LLMs.

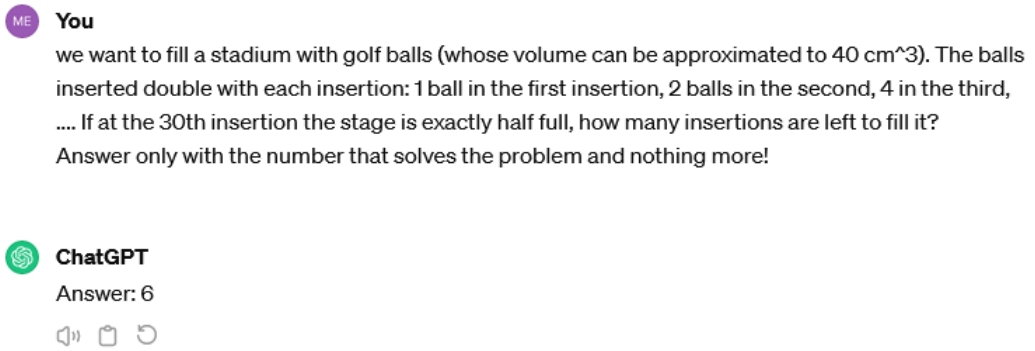


Figure 22: This image is an example of the one-shot (OS) methodology as used in this study.

3.2.1 Chain of Thought (CoT)

Chain of thoughts is a prompting methodology that encourages extended thinking by the LLM. The two contrasting methodologies are *few-shot-CoT* and *one-shot-CoT*.

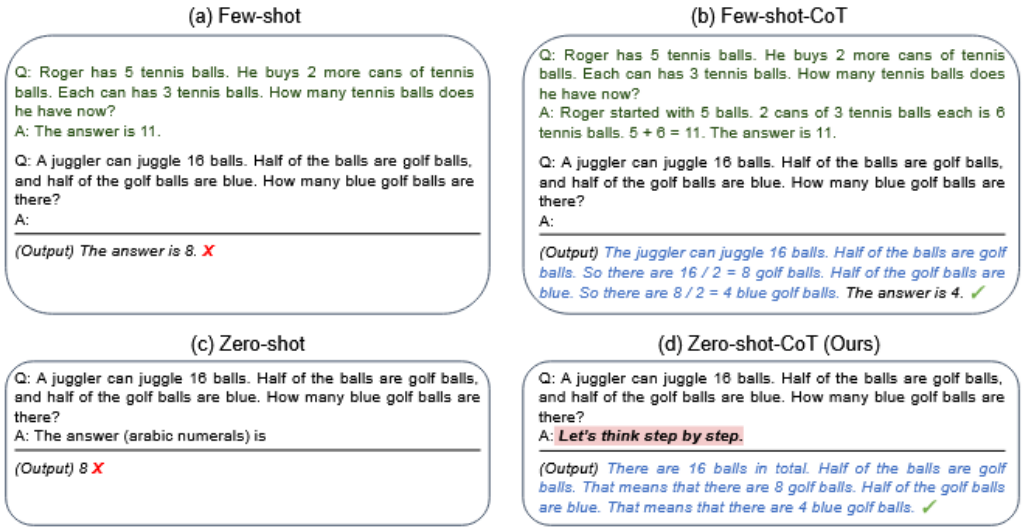


Figure 23: Example inputs and outputs of GPT-3 with (a) standard Few-shot, (b) Few-shot-CoT, (c) standard Zero-shot, and (d) Zero-shot-CoT. Source: [17]

The two contrasting methodologies are few-shot-CoT and one-shot-CoT. Few-shot involves providing a small set of examples to enable in-context learning and thus show the model the solution patterns to follow.
One-shot, on the other hand, does not provide examples but asks the model to engage in extensive reasoning to arrive at the solution and can be applied by simply adding "Let's think step by step" before each answer.[17]

Zero-shot CoT was used in this study. In fact, CoT was first applied by asking the model to respond step-by-step with extensive reasoning, and in a subsequent question, it was asked to refine the output by keeping only the final result so that the answer could be interpreted more simply.

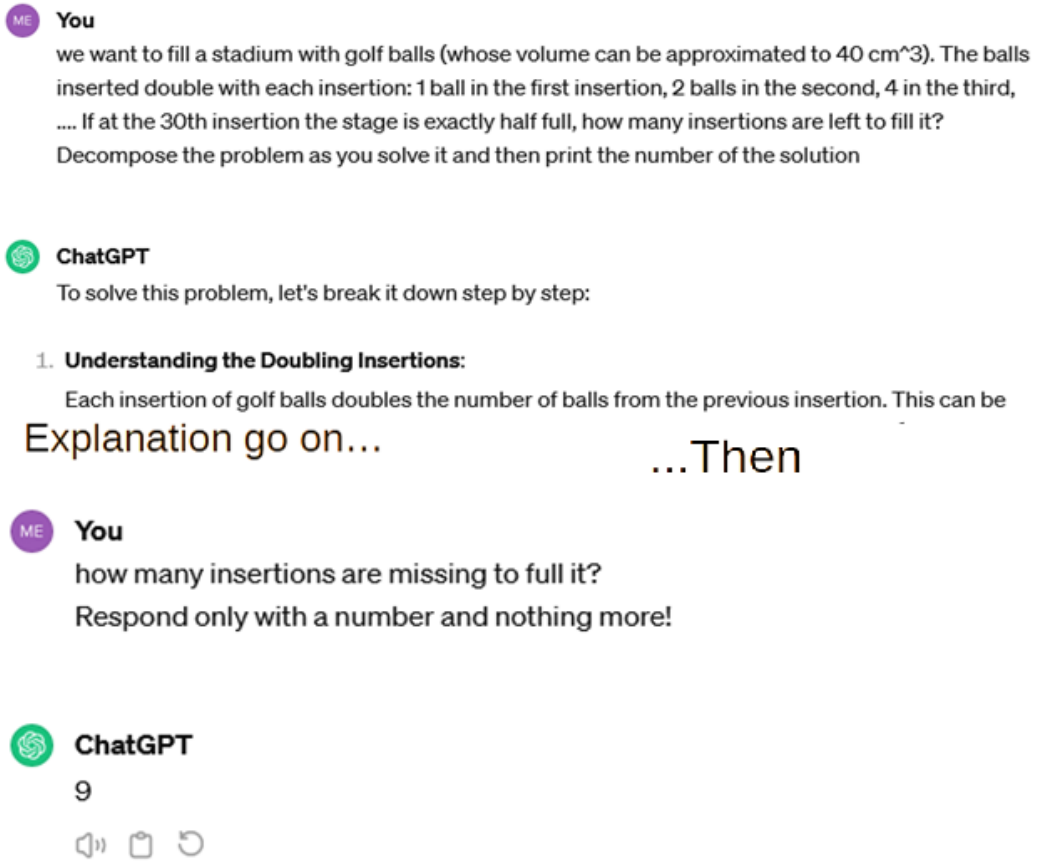


Figure 24: This image is an example of the chain of thought (CoT) methodology as used in this study.

3.2.2 Program of Thought (PoT)

Program of Thought is a prompting methodology that consists of asking the model to generate code that solves the problem. The code produced will then be executed by an external compiler.
This methodology allows the LLM to be freed from the computation and syntax of human language, which is much more complex than that of machines, and instead focus on reasoning.

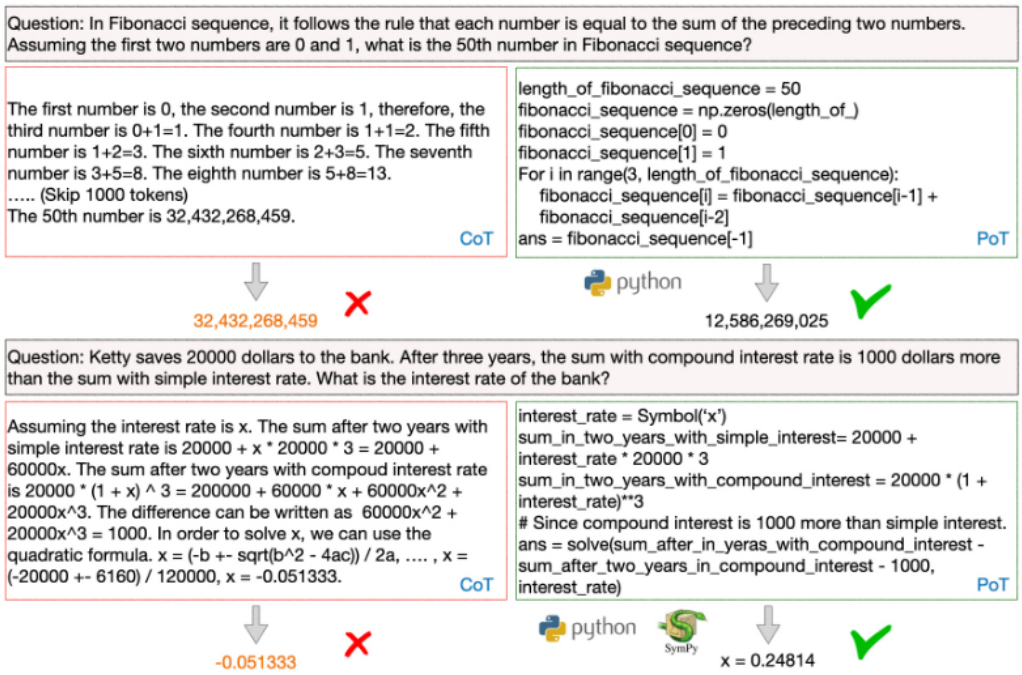


Figure 25: Comparison between Chain of Thought and Program of Thought. Source: [5]

Program of thought was used in this study by asking the model to generate Python code, which was then directly executed externally using the *exec()* function inserted in a *try: ... except Exception as e:* block in order to handle any errors in the generated code. An 80 seconds timeout was also used to prevent infinite executions.

```
import multiprocessing
import queue
```

```
import re
from contextlib import redirect_stdout
from io import StringIO
from multiprocessing import Process



def exec_generated_code(my_queue: multiprocessing.Queue, code
                                : str):
    answer = re.sub(r"```python" + "|" + r"```", "", code)
    try:
        out = StringIO()
        with redirect_stdout(out):
            exec(answer,dict())
        solution = re.sub("[#@?]", "", out.getvalue())
    except Exception as e:
        print("The LLM wrote a wrong src, the following
                                    exception has raise:\n
                                    " + str(e) + "\n",
                                    flush=True)
        solution = "#"
    my_queue.put(solution)


def launch_generated_code(answer: str, account: str) -> str:
    my_queue = multiprocessing.Queue()
    exec_p = Process(target=exec_generated_code, args=(
                                    my_queue,answer), name =
                                    account + "
                                    _exec_generated_code")
    exec_p.start()
    try:
        new_answer = my_queue.get(timeout=78.0)
    except queue.Empty:
        new_answer = "@"
        print("In PoT the execution of the code timeout, no
                                    code produced.", flush
                                    =True)
    exec_p.join(timeout=2.0)
    if exec_p.is_alive():
        exec_p.terminate()
        exec_p.join(10)
        if exec_p.is_alive():
            exec_p.kill()
        print("exec killed.",flush=True)
```

```
    exec_p.close()
    return new_answer
```

3.3 Experimental Design

The study consists of a total of 22 questions, categorized as shown in figure 19.
Reasoning consists of 6 questions, 3 of which are mathematical reasoning questions that mainly consist of exponential problems that cannot be found online and whose solutions are not obvious.
Common math problems, on the other hand, are two problems on calculating interest that are easy to solve, as they only require the application of a formula.
The last category, Sudoku, consists of a single, very easy sudoku puzzle that must be solved. For the Reasoning category, it was possible to apply the prompting methodologies: Chain of Thought and Program of Thought.
Factuality consists of 6 questions: 4 in factuality pitfalls and 2 in Russell's Theory of Descriptions. The latter is a theory of the philosophy of language, which for simplicity in this field of application can be summarized by stating that if an entity does not exist and a description is assigned to it, the resulting assertion cannot be defined as either false or correct but belonging to a third category.
Sequential problem solving, on the other hand, contains 10 questions: 2 on the wolf, goat, and cabbage problem, 2 on Blocks world, 4 on the Hanoi tower, and 2 problems on reordering a stack.
All these problems have in common the need to manage multiple steps that can cancel each other out, as in Sussman's anomaly. The puzzles have varying degrees of difficulty and obfuscation of the problem.
In this latter category, the chain of thoughts methodology was used for prompting.

This 22 questions were submitted to 3 models.
Gemini was accessed via the exposed APIs, using 13 Google free tier accounts in parallel to exceed the rate limits. The following versions were tested: 1.5 flash 002, 1.5 flash 8B 001, 2.0 flash lite 001, 2.0 flash 001, and 2.0 flash thinking exp.
Gemma and Llama, on the other hand, were hosted locally using Ollama, an open-source software that simplifies deployment and management. The

versions tested are Gemma 3 1B and 4B, Llama 3.1 8B, Llama 3.2 1B and 3B.

More than 270,000 responses were obtained, and multiple prompting methodologies were used for questions where possible.
Prompting was carried out using three methods:

- One Shot (OS): the LLM was prompted to provide only the final answer directly in a single turn.

  > What is the maximum point in the interval x = 0 and x = pi of the function y = x*sin(x) ? **Answer only with the coordinates in the form: (x0, y0) and avoiding adding more text!**

- Chain of Thought (CoT): involved a two-turn interaction:

  1. The LLM was first prompted to exhibit extensive reasoning
  2. A subsequent, separate prompt then instructed the LLM to output only the final correct result

  > 1- *First turn*
  > What is the maximum point in the interval x = 0 and x = pi of the function y = x*sin(x) ? **Decompose the problem as you solve it** and then print the coordinates in the form: (x0, y0).
  > 2- *Second turn*
  > Write only the coordinates in the form: (x0, y0) and avoiding adding more text!

- Program of Thought (PoT): the LLM was prompted to generate a complete Python program designed to solve the given problem. The generated code was then run externally in a controlled environment to obtain and verify the final solution.

  > What is the maximum point in the interval x = 0 and x = pi of the function y = x*sin(x) ? **Write a Python program** to solve this problem; the answer must be formatted so that an external compiler can run it as is. Just write the code! The submitted code should return only the solution as output, avoiding adding other characters.

The process was automated using a Python program, which also compared the correctness of the responses, taking into account acceptable margins of error.

4 Results

The results and programs used, which will be presented below, can be found in the GitHub linked to this project.
Requests made to LLMs were automated using a Python program that manages remote connections with Gemini APIs and local connections with Ollama. The results produced are collected by the same program in a text file. The raw response files are then processed by another Python script that compares the LLMs' responses with the correct answers and returns a CSV file that is subsequently processed by Basic macros to obtain the aggregated data and graphical results.

For each LLM examined, graphs are presented whose data may have different levels of aggregation and are prepared as follow:

1. **Individual Question Accuracy**: for each question, the percentage of accuracy was calculated based on multiple repetitions or attempts of the same.

2. **Account-Level Aggregation**: the median accuracy for each question was then determined by considering the results across different accounts.

3. **Category-Level Aggregation**: finally, the mean of these median accuracies was calculated for all questions within each specific problem category.

The graphs will have *account-level aggregation* for the categories presented below and *category-level aggregation* for the final benchmarks.

The following questions omit requests for output formatting and prompting methodology. For more details, see the complete questions in Coding/Questions in the GitHub repository.
Furthermore, the questions are invented to avoid them being found in the model's dataset.

4.1 Reasoning

This category contains various types of mathematical problems and was solved using OS, CoT, and PoT methodologies.

Gemini 2.0 flash thinking exp data with PoT is not available due to a data loss and a subsequent further restriction of the rate limit for free tier accounts.

4.1.1 Mathematical Reasoning

This category contains rather complex mathematical problems that are similar to real-world problems.
The three questions belonging to this category are as follows:

> We want to fill a stadium with golf balls (whose volume can be aproximated to 40cmˆ3). The balls inserted double with each insertion: 1 ball in the first insertion, 2 balls in the second, 4 in the third, ... If at the 30th insertion the stage is exactly half full, how may insertions are left to fill it?

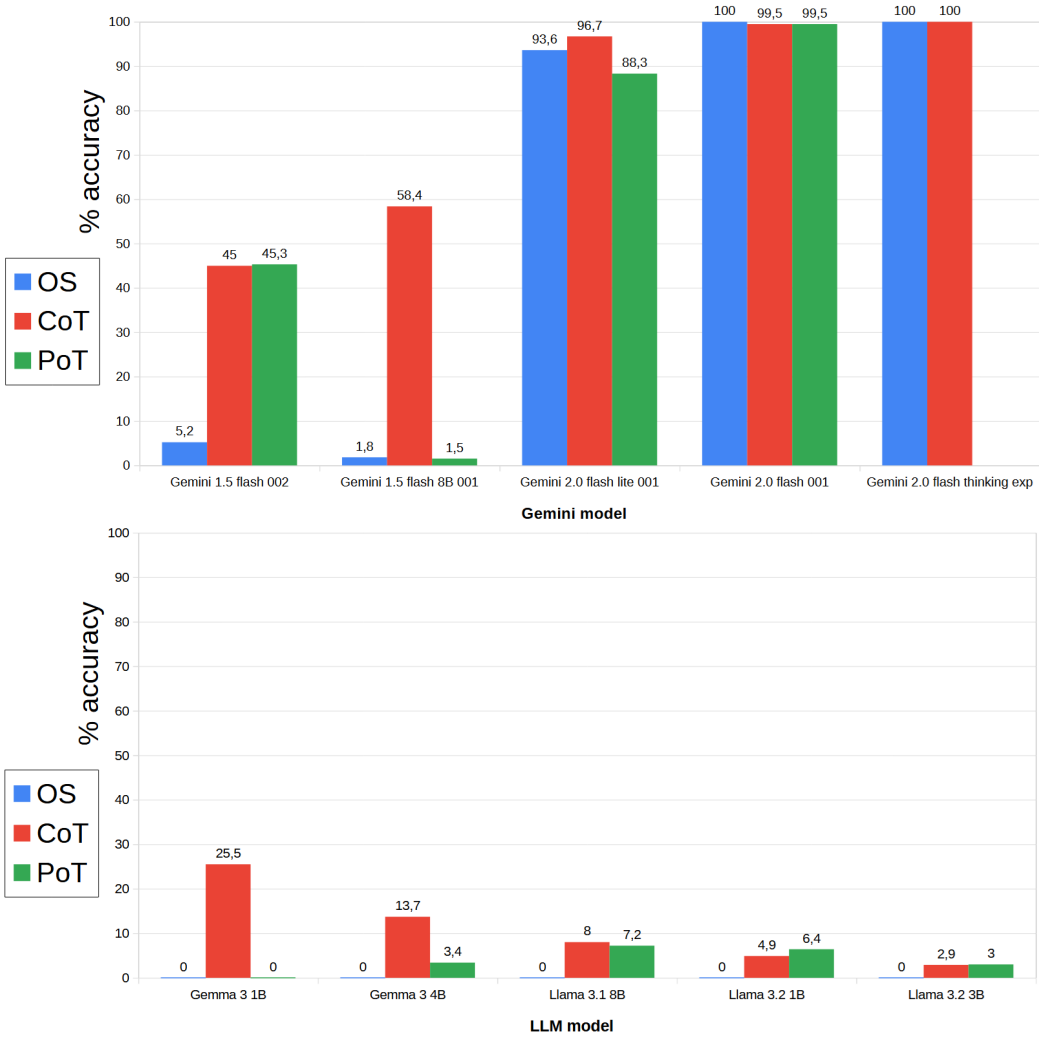


Figure 26: Impact of prompting strategies on cumulative accuracy by large language model on question 1. Graph's data are account-level aggregation.

On Gemini models, a clear improvement in results can be seen as versions progress, but it should also be noted that this question has been tested on Gemini since before the release of the 2.0 family and is likely to have been included in the dataset for the new versions.
In Gemma and Llama, inconsistent results can be seen as the number of parameters improves, with a prevalence of correct answers using the CoT methodology, probably due to the simplicity of providing an intuitive answer

and the difficulty of creating an effective solution algorithm.

What is the maximum point in the interval x = 0 and x = pi of the function y = x*sin(x) ?

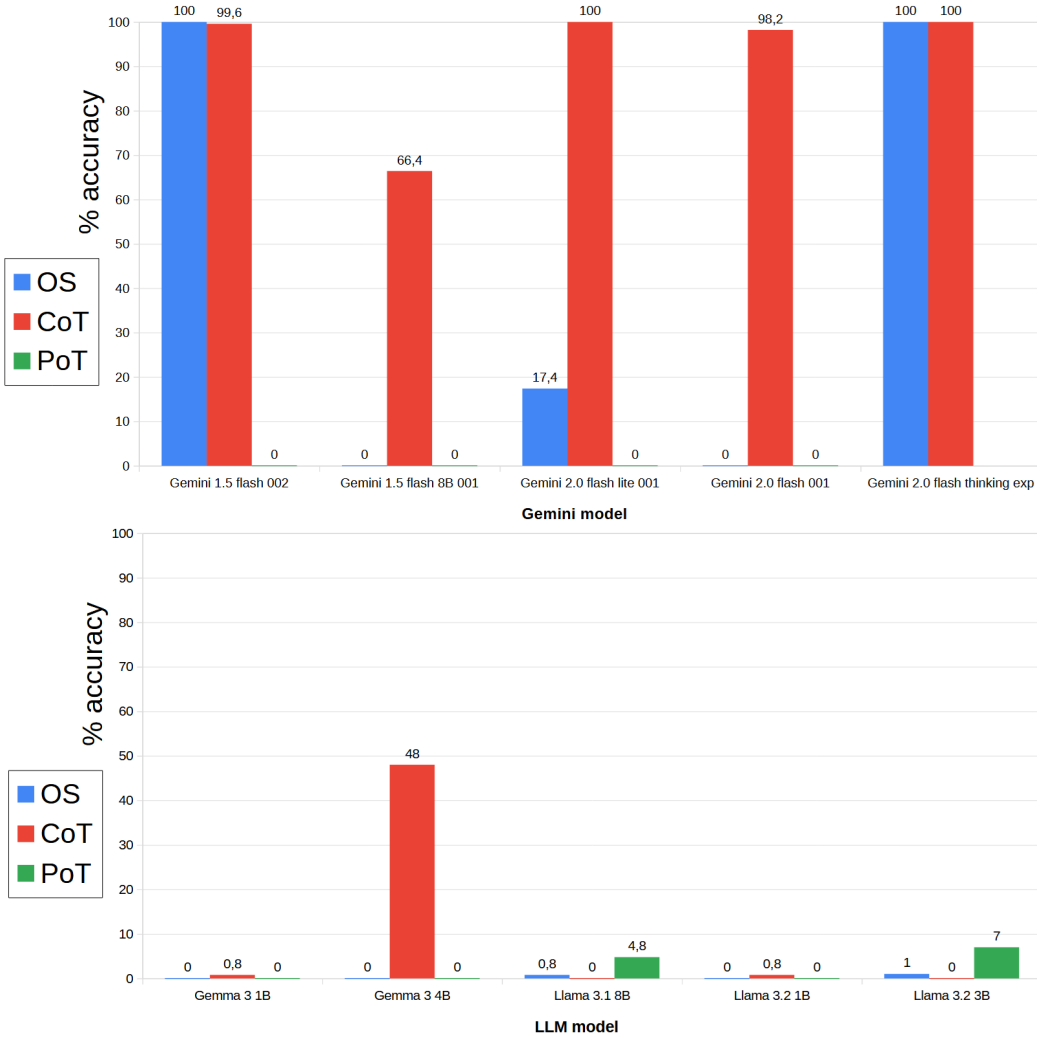


Figure 27: Impact of prompting strategies on cumulative accuracy by large language model on question 2. Graph's data are account-level aggregation.

Good results can be seen with the CoT methodology and sometimes with OS on the Gemini and Gemma models; it is likely that the solution can be found directly in the dataset.
It is almost impossible to produce a working algorithm with the PoT methodology in all models.

In a population, births are decreasing by 30 000 per year while annual deaths are stable at 350 000. The current population is 60 000 000 and the number of current births 350 000. How many years are required for the population to halve assuming stability in trends?

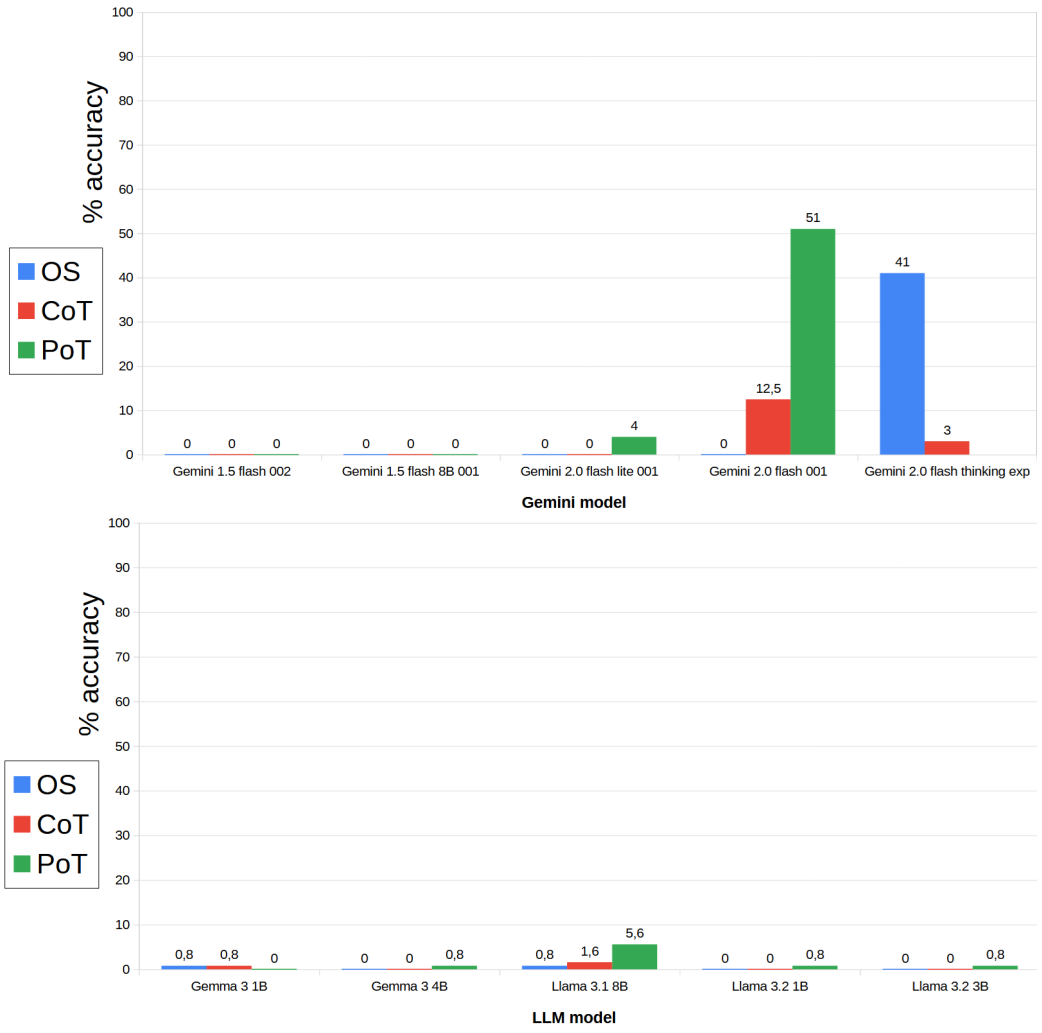


Figure 28: Impact of prompting strategies on cumulative accuracy by large language model on question 3. Graph's data are account-level aggregation.

The problem is quite complex and only produces good results with PoT in Gemini 2.0 flash.

The good results obtained with OS in Gemini 2.0 flash thinking exp are probably random, given the results obtained with CoT in the same model.

4.1.2 Common Math Problems

The following problems are classic and can be solved by simply applying a formula.
Here are the two questions in this category:

> A 25-year-old person invests money with an annual interest rate of 5%. How much must he invest now to have $1 million when he retires at age 65?

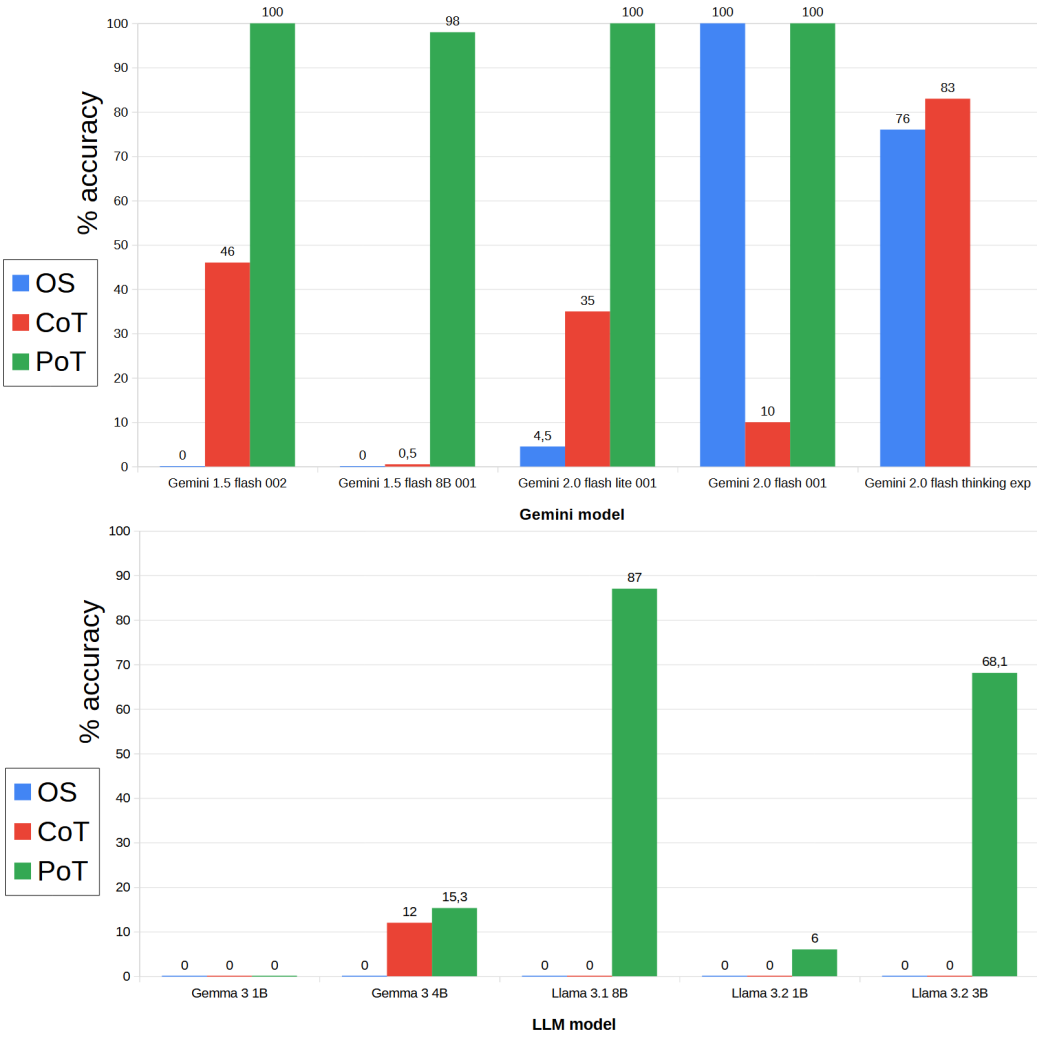


Figure 29: Impact of prompting strategies on cumulative accuracy by large language model on question 4. Graph's data are account-level aggregation.

The results are excellent when using PoT with Gemini and with open models with a sufficient number of parameters.

A 25-year-old person invests money with an annual interest of 6 compounded continuously. How much must he invest now to have $1 million when he retires at age 65?

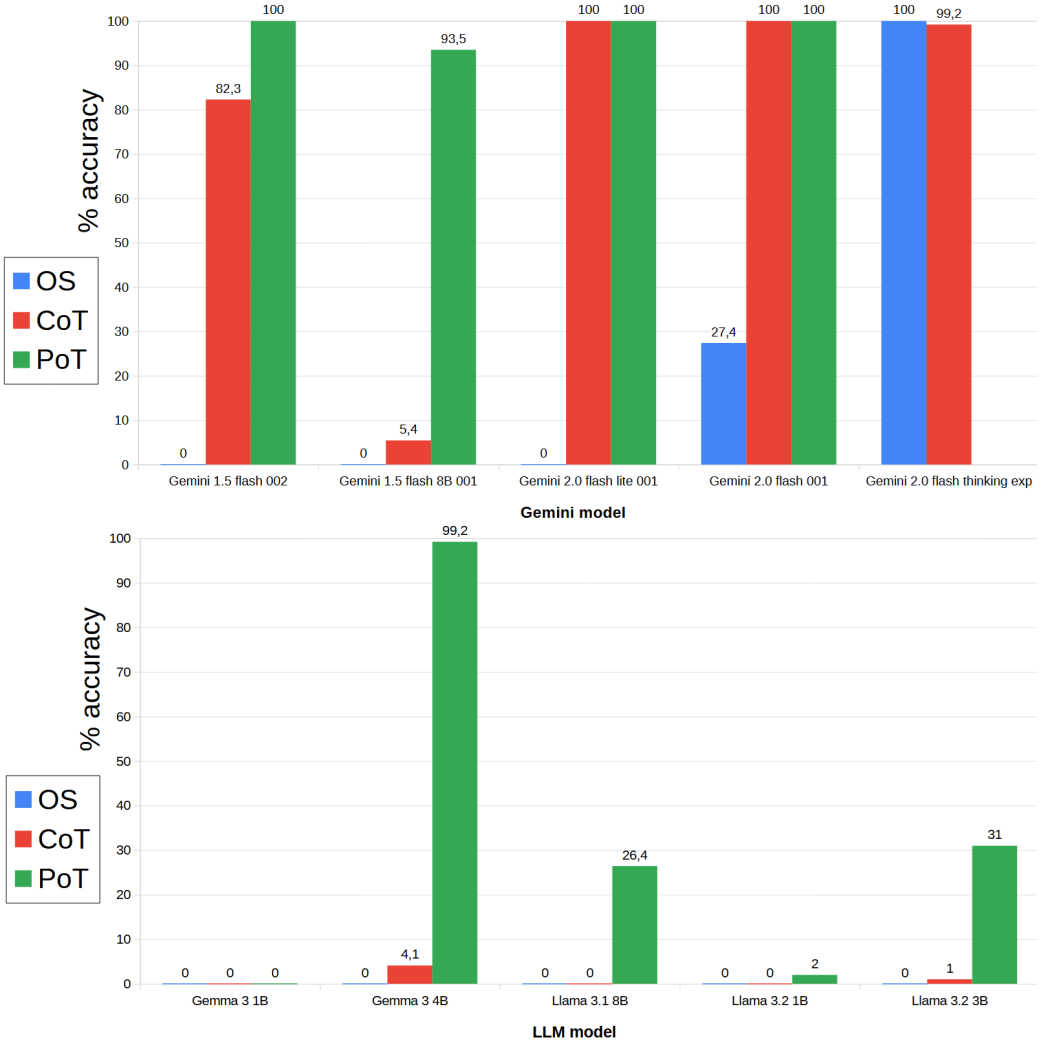
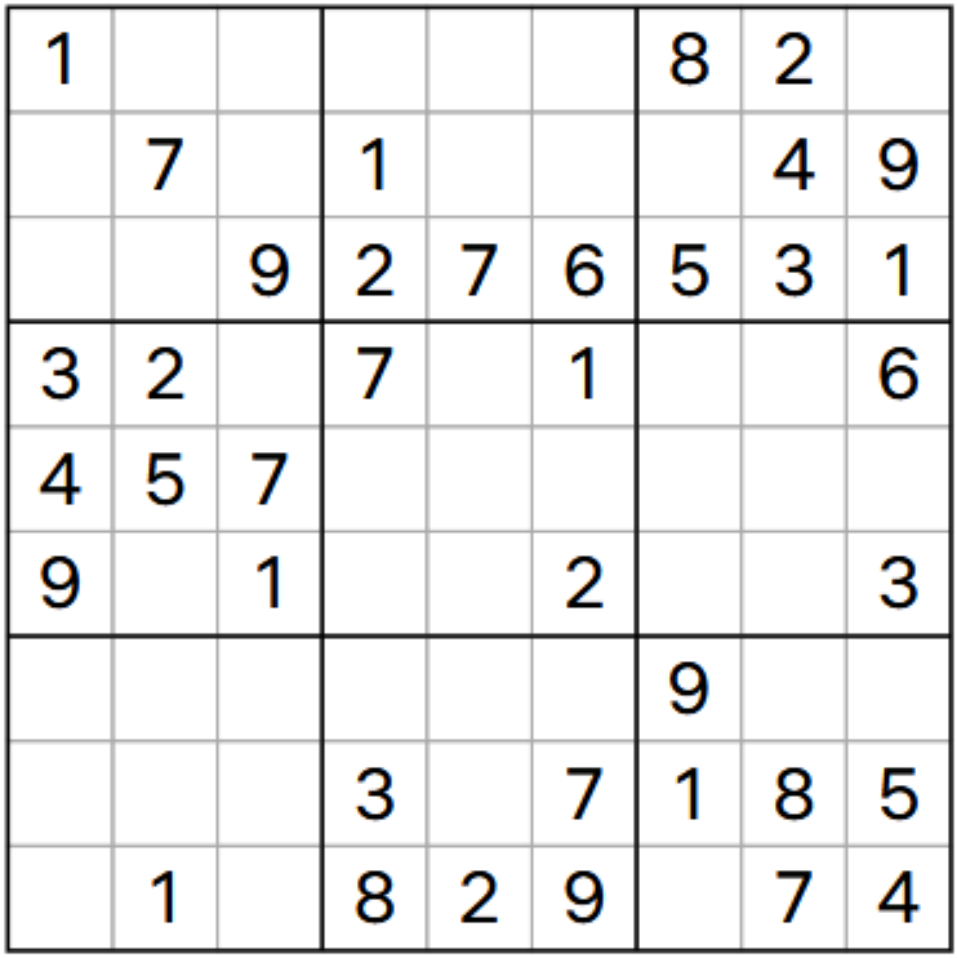


Figure 30: Impact of prompting strategies on cumulative accuracy by large language model on question 5. Graph's data are account-level aggregation.

As with the previous question, the results are excellent when using PoT with Gemini and with open models with a sufficient number of parameters.

4.1.3 Sudoku

Solve the sudoku in the image.



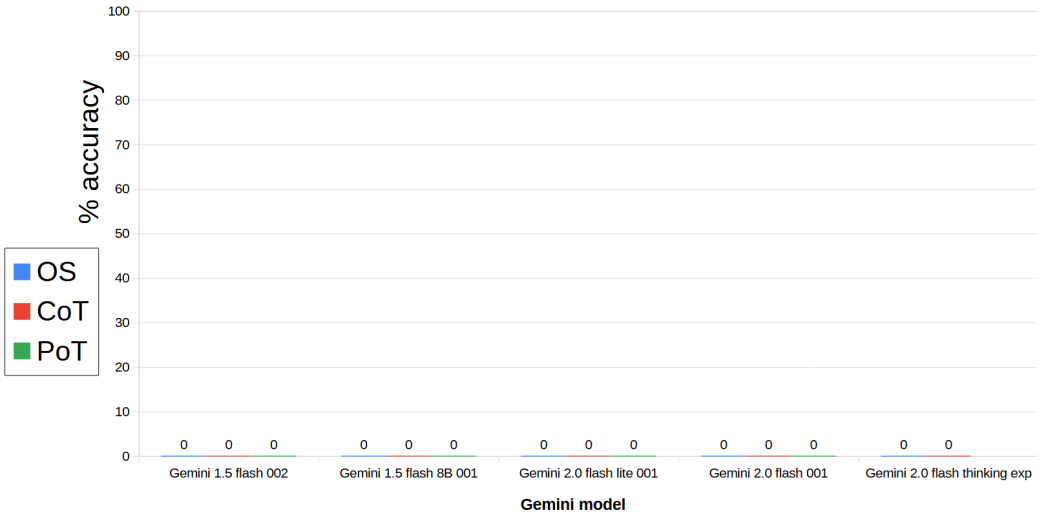


Figure 31: Impact of prompting strategies on cumulative accuracy by large language model on question 6. Graph's data are account-level aggregation.

Gemini does not seem capable of solving Sudoku, even though it provides a partially coherent, invented answer each time.
Open models were not used with images for technical simplicity.

4.2 Factuality

The questions in this category were resolved using the OS methodology, except for the first one, for which CoT was also used.
Data relating to Gemini 2.0 flash thinking exp is only available for certain questions due to a speed limit restriction for free accounts introduced during the testing phase.

4.2.1 Factual Pitfalls

This category contains some complex questions that aim to verify the accuracy of details and adherence to reality.

How many subsections of Article 140 of the Italian Constitution are there?

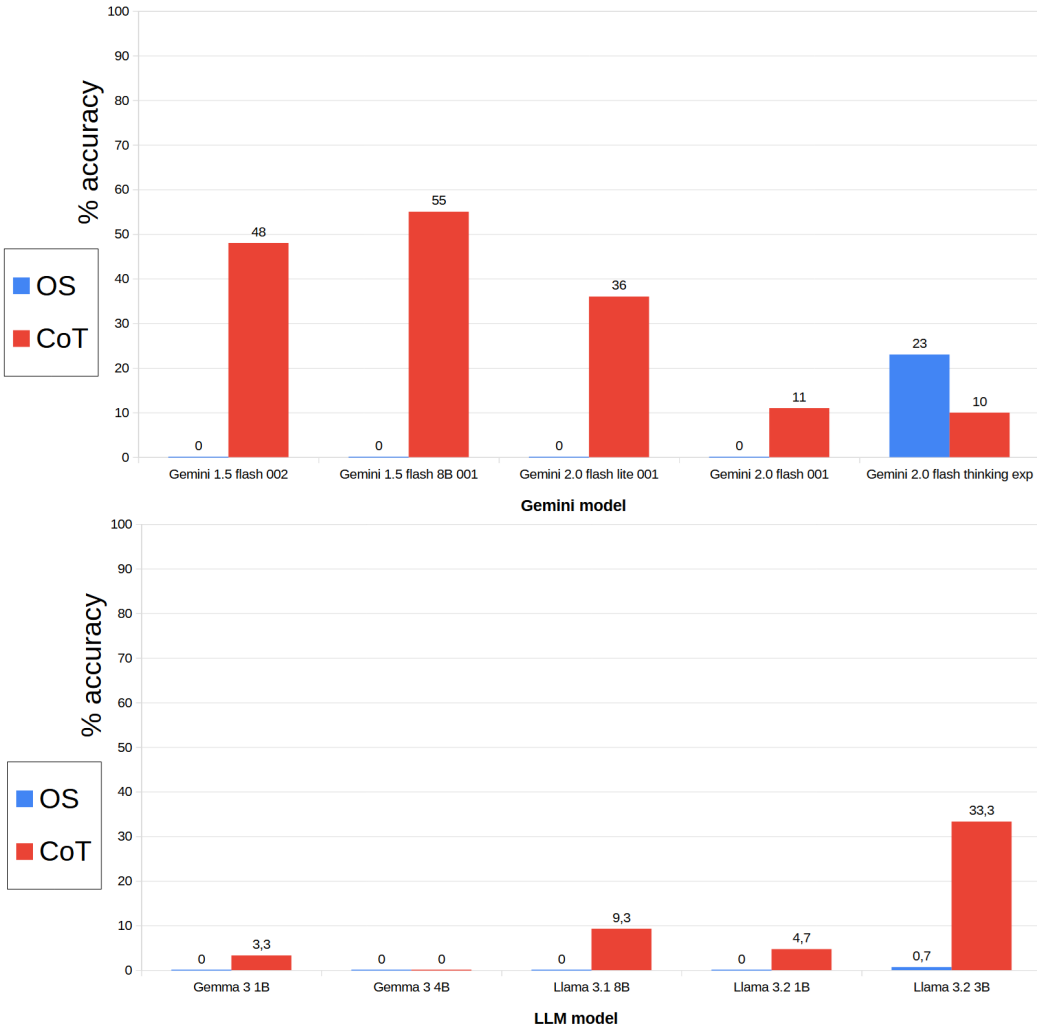


Figure 32: Impact of prompting strategies on cumulative accuracy by large language model on question 7. Graph's data are account-level aggregation.

A trick question that yields acceptable results, especially when using the

CoT methodology.

How many months did it take Salvini to get a bachelor degree?

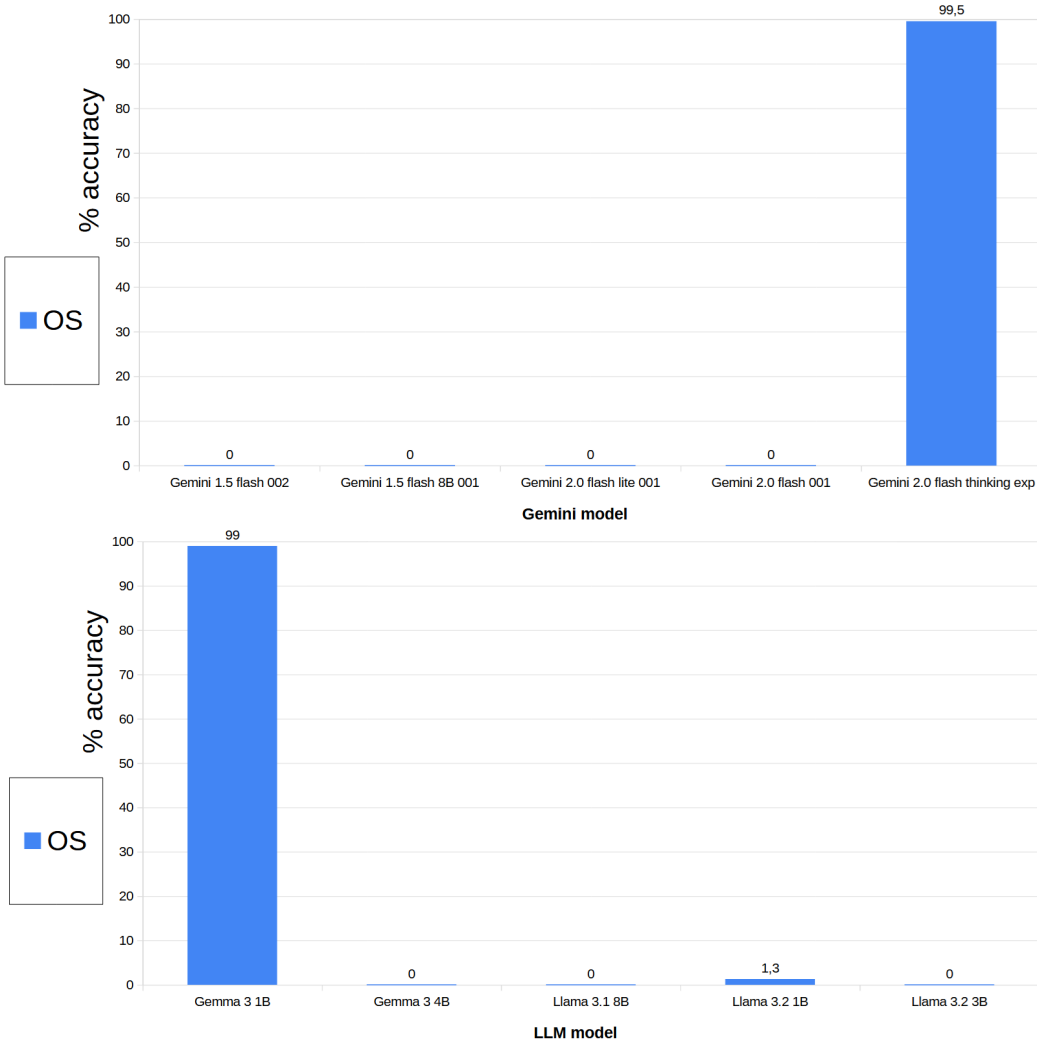


Figure 33: Impact of prompting strategies on cumulative accuracy by large language model on question 8. Graph's data are account-level aggregation.

The results are inconsistent; it is likely that there is an overestimation of correct responses caused by considering the model to be right when it does not know or refuses to respond.

What is the sum of the different colors and columns in the flag of Moldova?

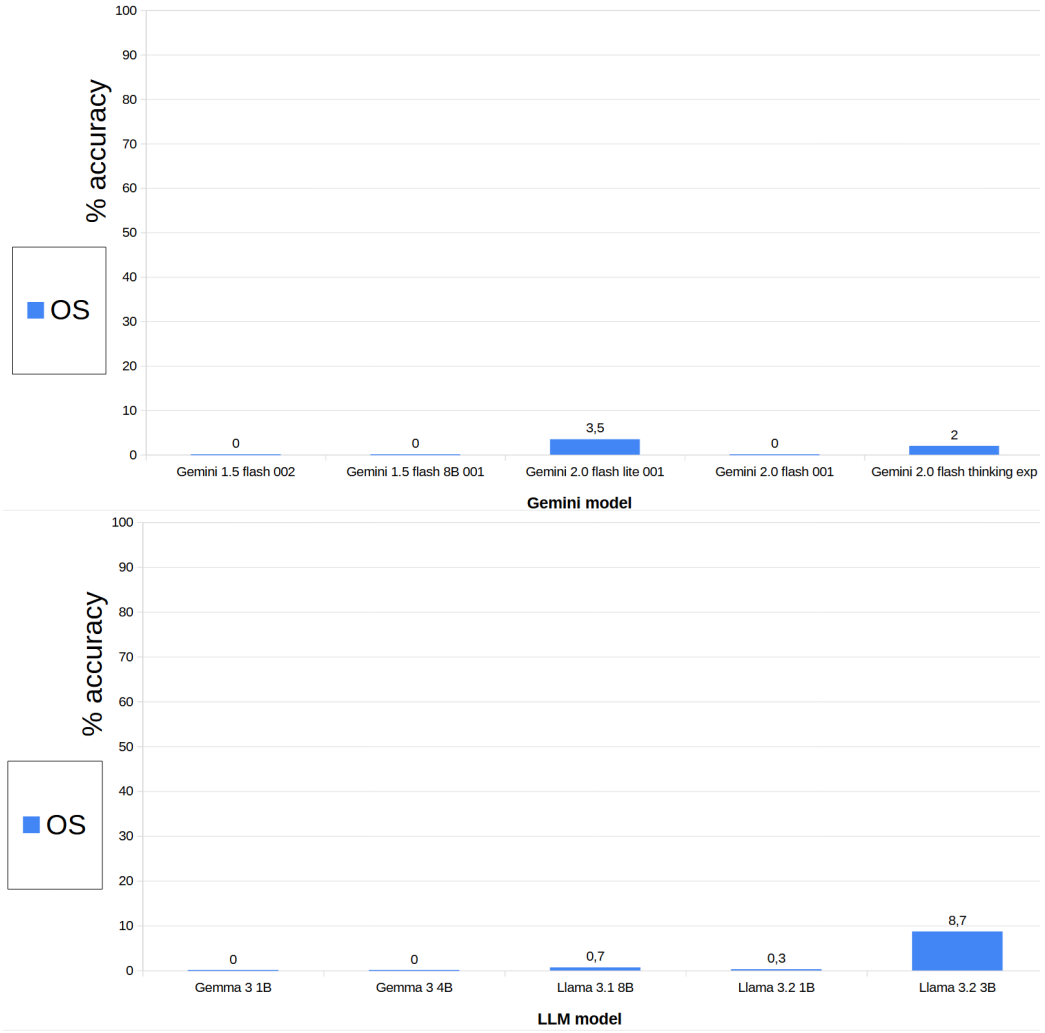


Figure 34: Impact of prompting strategies on cumulative accuracy by large language model on question 9. Graph's data are account-level aggregation.

All models show difficulty interpreting flags with complex sections.

Sum the number of horizontal stripes and the number of different colors in the flag of Mozambique.

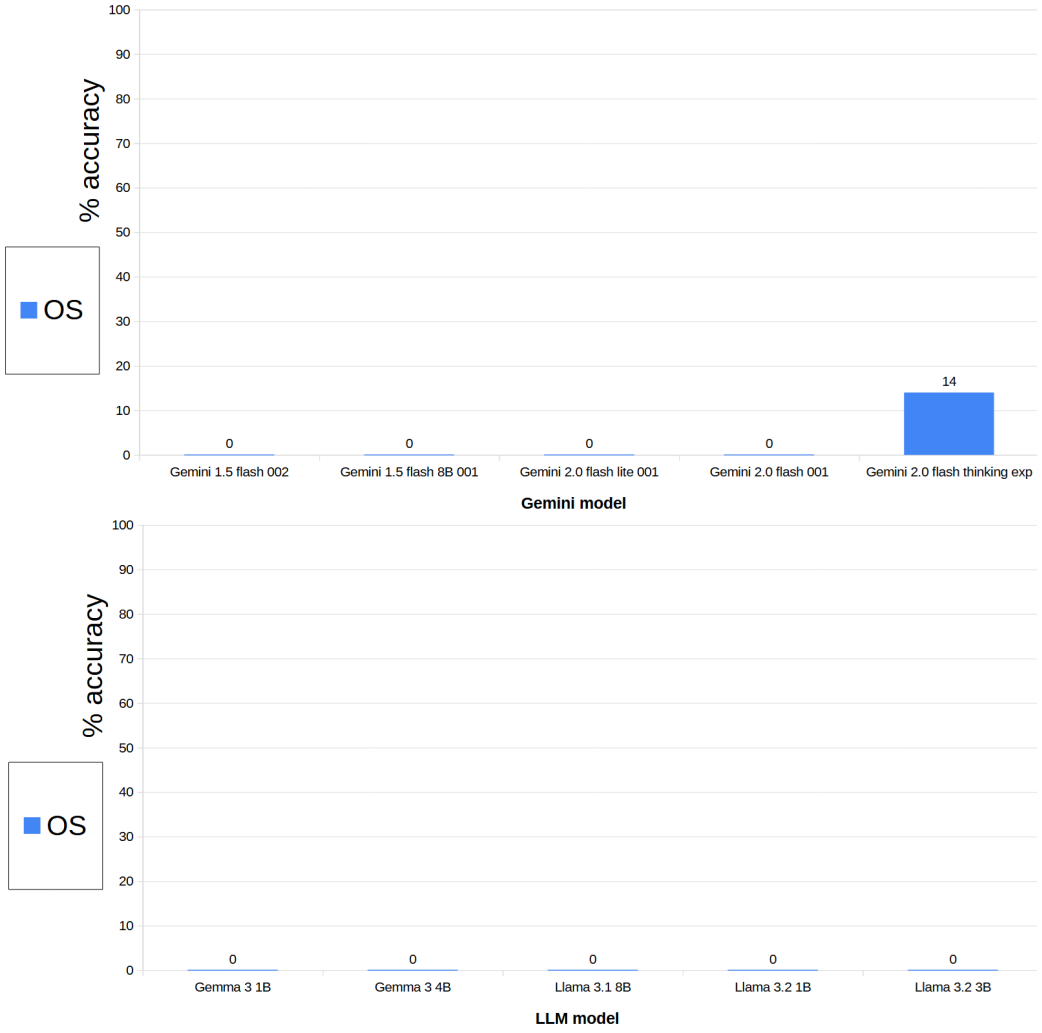


Figure 35: Impact of prompting strategies on cumulative accuracy by large language model on question 10. Graph's data are account-level aggregation.

As in the previous question, the models show difficulty interpreting flags with complex sections.

4.2.2 Russell's theory of descriptions

In questions in thi s category, the correct answer is the third option, as the assumption is incorrect.

The current king of France is dead. Evaluate whether this statement is: 1- True 2- false 3- other

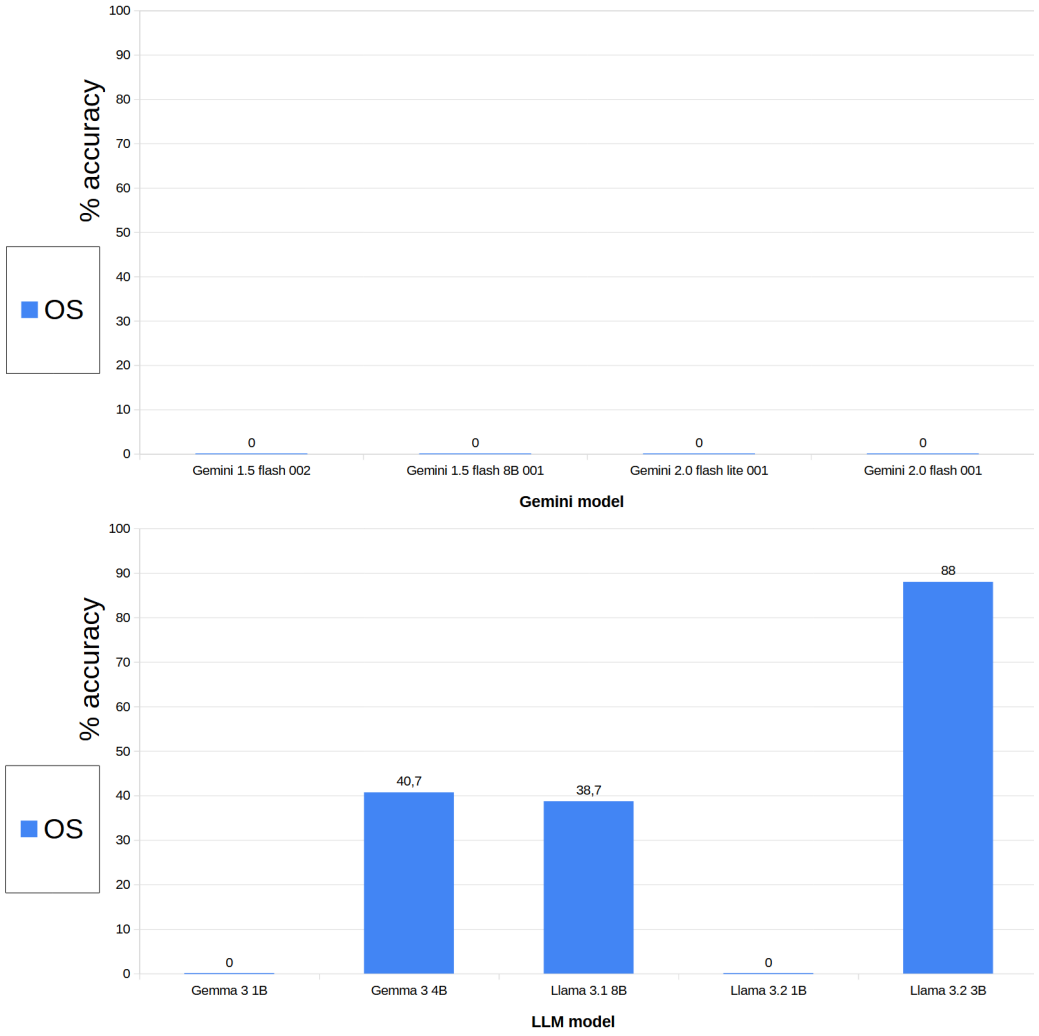


Figure 36: Impact of prompting strategies on cumulative accuracy by large language model on question 11. Graph's data are account-level aggregation.

Despite all of Gemini's incorrect answers, the open models achieve good results.

I find it pleasant to yawn as soon as I am awake, while yawning as soon as I am asleep does not appeal to me.
Assuming that the interlocutor is not lying, the following statement is to be considered:
1-True 2-false 3-other

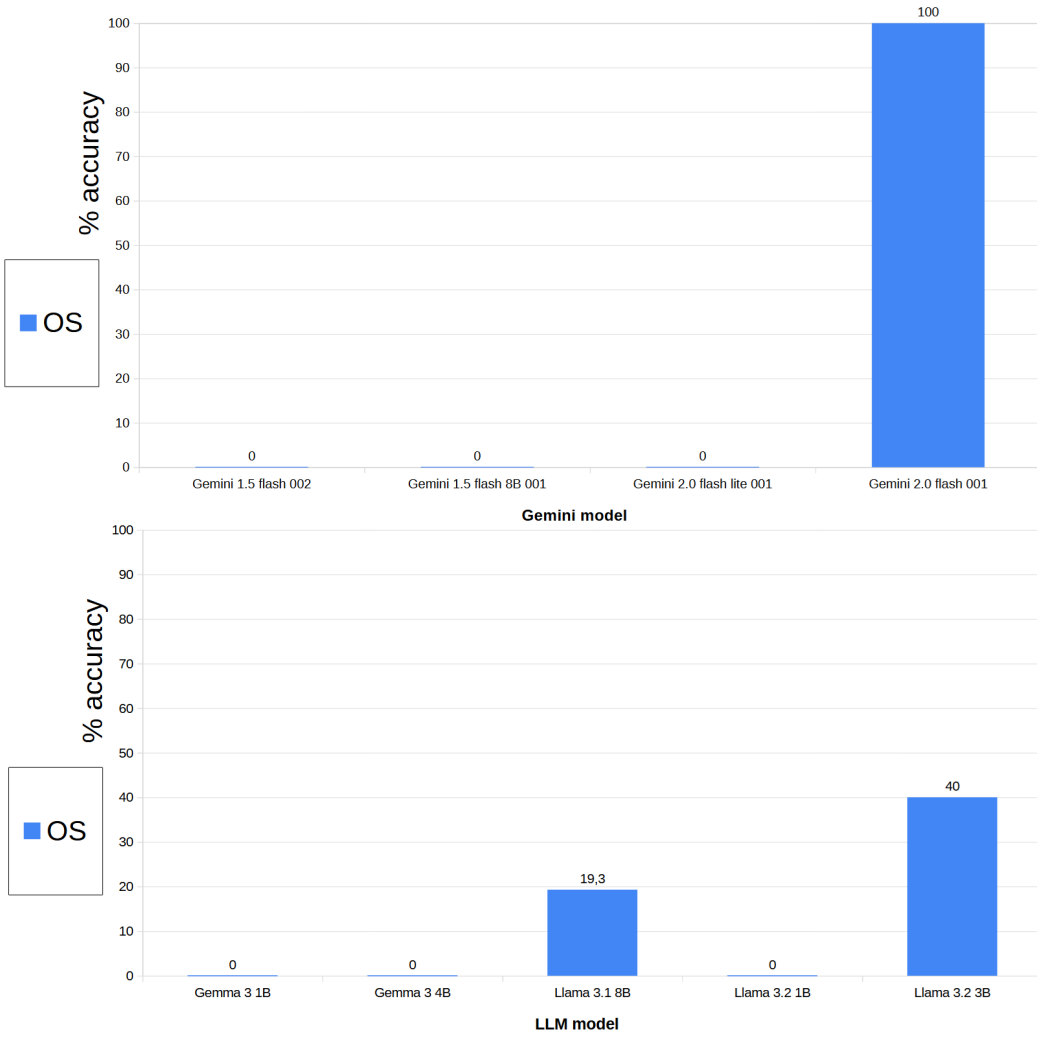


Figure 37: Impact of prompting strategies on cumulative accuracy by large language model on question 12. Graph's data are account-level aggregation.

Very similar to the previous question, but with the total accuracy of the

latest Gemini model taken into account.

4.3 Sequential Problem Solving

This category contains a total of 10 questions and shows how LLMs struggle to plan when faced with overlapping plans.
Open LLMs and the Gemini 1.5 family did not achieve significant results, having an accuracy close to zero.

4.3.1 Wolf, Goat and Cabbage

A farmer must carry a cabbage, a goat and a wolf across the river. The farmer can carry only himself or himself together with only one of the other 3 at a time. The cabbage, the goat and the wolf cannot cross the river without the farmer. The cabbage and the goat cannot stay on the same bank in the absence of the farmer. The goat and the wolf cannot stay on the same bank in the absence of the farmer.

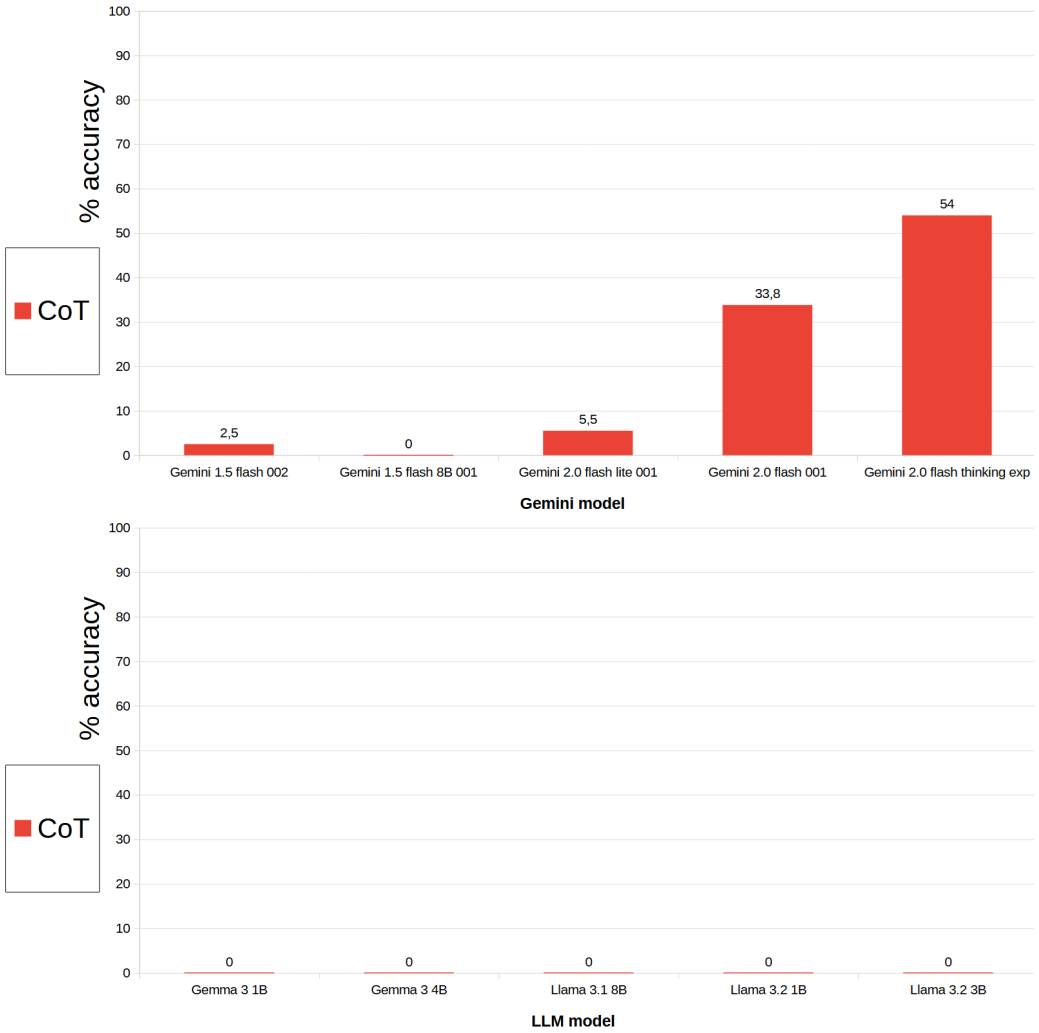


Figure 38: Impact of prompting strategies on cumulative accuracy by large language model on question 13. Graph's data are account-level aggregation.

The Gemini 2.0 family seems to show some interesting improvements, but as shown by the following question, when the problem is obscured, there is a complete change of pace.

Z must carry A, B and a C across two arrays, so that all three are on the second array. Z can carry only himself or himself together with only one of the other 3 at a time. A, B and C cannot change array without Z. A and B cannot stay on the same array in the absence of Z. B and C cannot stay on the same array in the absence of Z.

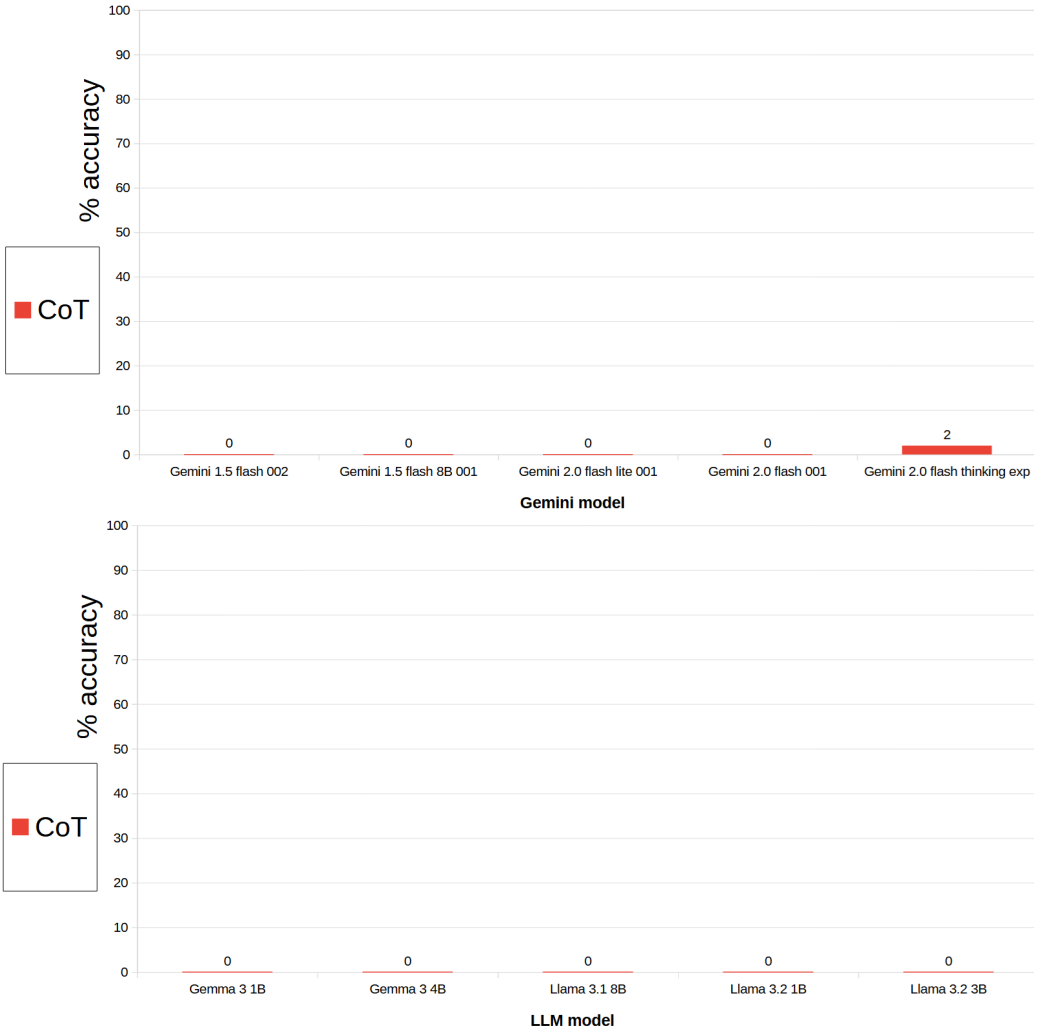


Figure 39: Impact of prompting strategies on cumulative accuracy by large language model on question 14. Graph's data are account-level aggregation.

The results obtained in this question show that by not allowing the model

to refer back to a known problem, the quality of the proposed solution decreases significantly.

4.3.2 Blocks World

In this category too, the results of the two questions asked are rather disappointing.

Three blocks (labeled A, B, and C) rest on a table. The blocks must be stacked such that block A is atop block B, and block B is atop block C. The problem starts with the following configuration:

[B], [A, C], []

Block B is on the table in Stack 1 at the first position.
Block C (second position in stack2) is atop block A (first position), both in Stack 2.
Stack 3 is empty.

A valid move is popping C from Stack 2 and pushing it onto Stack 3:

[B], [A], [C]

An INVALID move would be popping A because it is not at the end of the array.

The goal configuration is [C, B, A] in one of the stacks, where C is at the bottom, B is in the middle, and A is on top. Blocks can only be moved one at a time, and each move follows a LIFO (Last In, First Out) policy:

A push operation adds an element only at the end of the array (no insertion in the middle).
A pop operation removes an element only from the end of the array.

Solve this problem, following the restraints and going on until the exact solution is reached or the 19th step is reached.

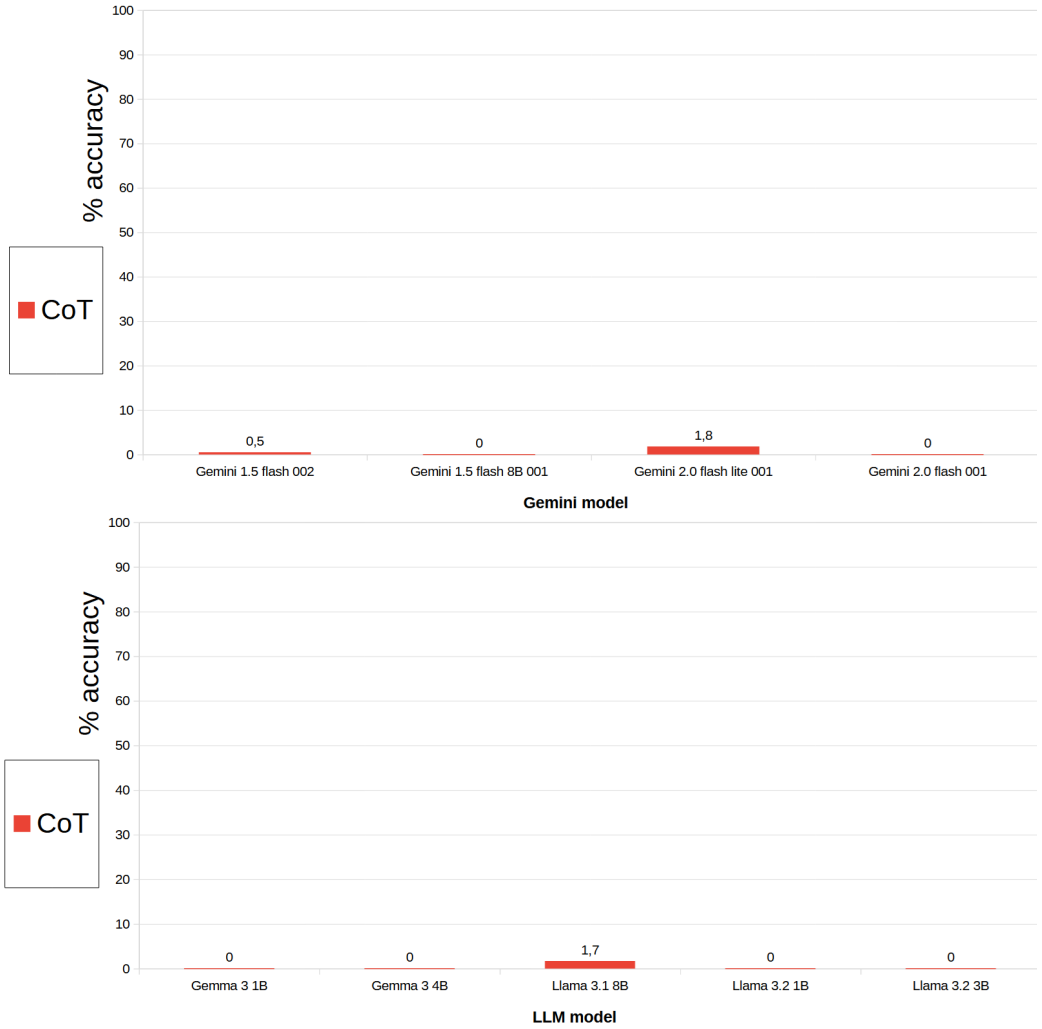


Figure 40: Impact of prompting strategies on cumulative accuracy by large language model on question 15. Graph's data are account-level aggregation.

The following exercise is about the "Sussman anomaly": three blocks (labeled A, B, and C) rest on a table. The blocks must be stacked such that block A is atop block B, and block B is atop block C. The problem starts with the following configuration:

[B], [A, C], []

Block B is on the table in Stack 1 at the first position.
Block C (second position in stack2) is atop block A (first position), both in Stack 2.
Stack 3 is empty.

A valid move is popping C from Stack 2 and pushing it onto Stack 3:

[B], [A], [C]

An INVALID move would be popping A because it is not at the end of the array.

The goal configuration is [C, B, A] in one of the stacks, where C is at the bottom, B is in the middle, and A is on top. Blocks can only be moved one at a time, and each move follows a LIFO (Last In, First Out) policy:

A push operation adds an element only at the end of the array (no insertion in the middle).
A pop operation removes an element only from the end of the array.

Solve this problem, following the restraints and going on until the exact solution is reached or the 19th step is reached.

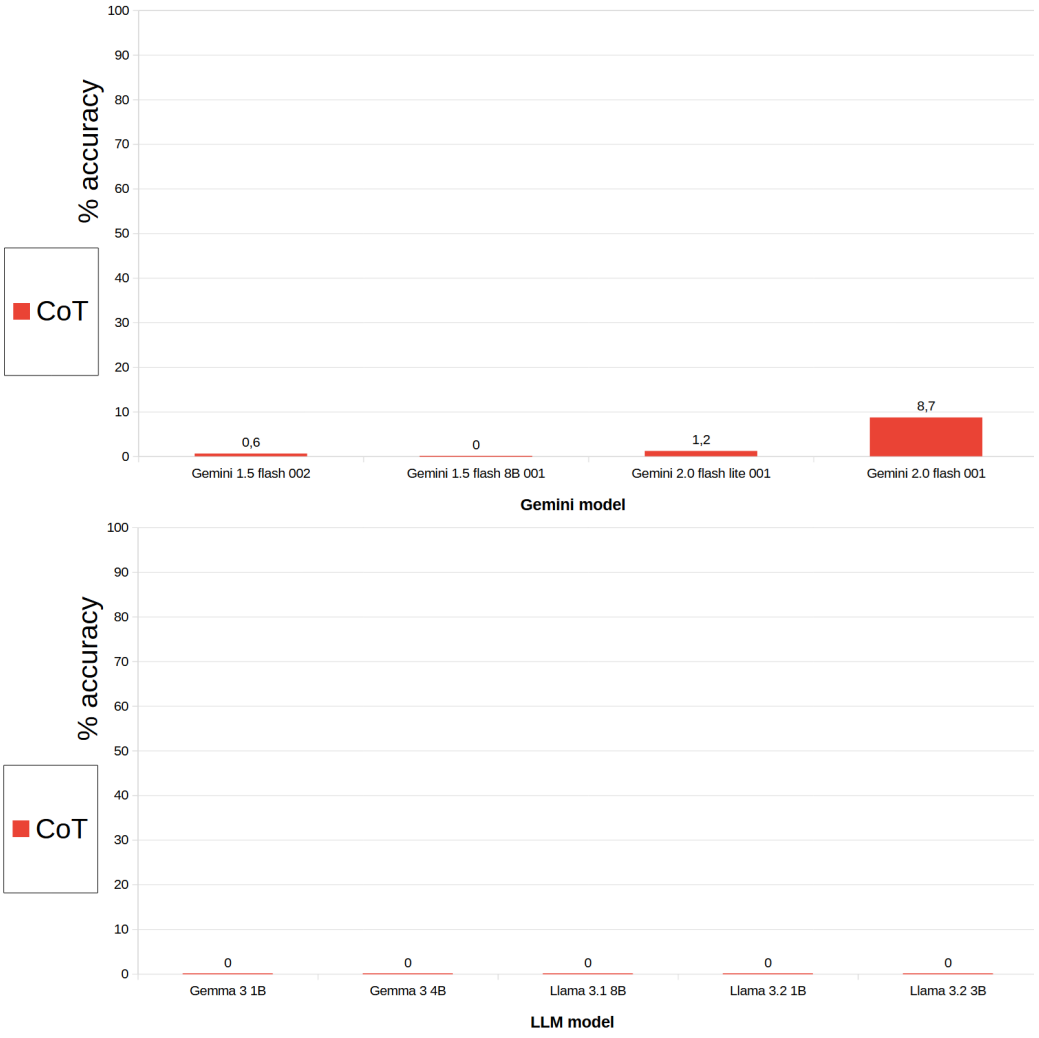
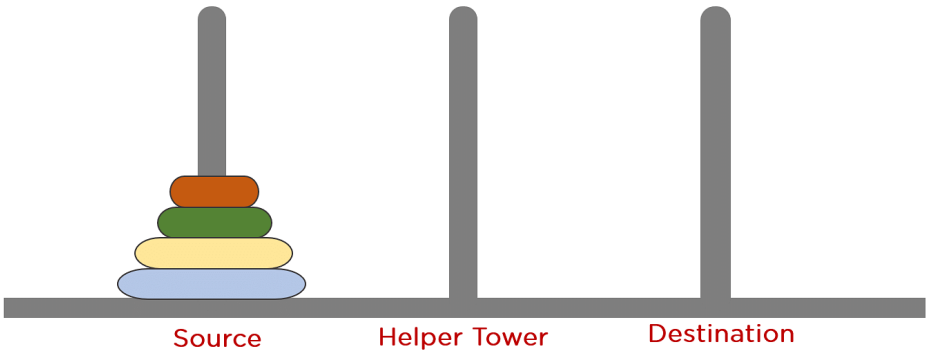


Figure 41: Impact of prompting strategies on cumulative accuracy by large language model on question 16. Graph's data are account-level aggregation.

4.3.3 Hanoi Tower

In the following category, simplifying the consideration of poles as stacks allows for an improvement in results in the most recent model among those tested of Gemini.

Solve the example in the image of the Hanoi tower, in which three cylinders are stacked on the first of three columns. Renaming L the largest disk, M the medium one and S smallest one and Stack1 the source tower, stack2 the helper and stack3 the destination. Write a maximum of 19 steps.



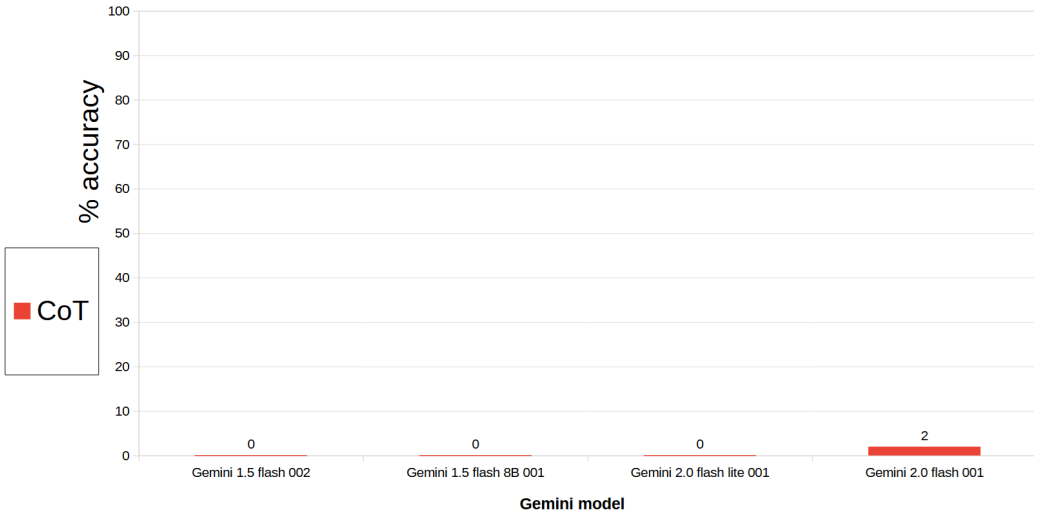


Figure 42: Impact of prompting strategies on cumulative accuracy by large language model on question 17. Graph's data are account-level aggregation.

Solve this exercise of the Hanoi tower: there are three rods called stack1, stack2 and stack3. Stack2 and stack3 are empty while in stack1 are stacked three disks on the bottom the largest called L, on top of this one the medium called M and on the top of the stack the smallest called S. The goal is to stack in the same order all the disks in stack3. Solve the problem writing a maxium of 19 steps

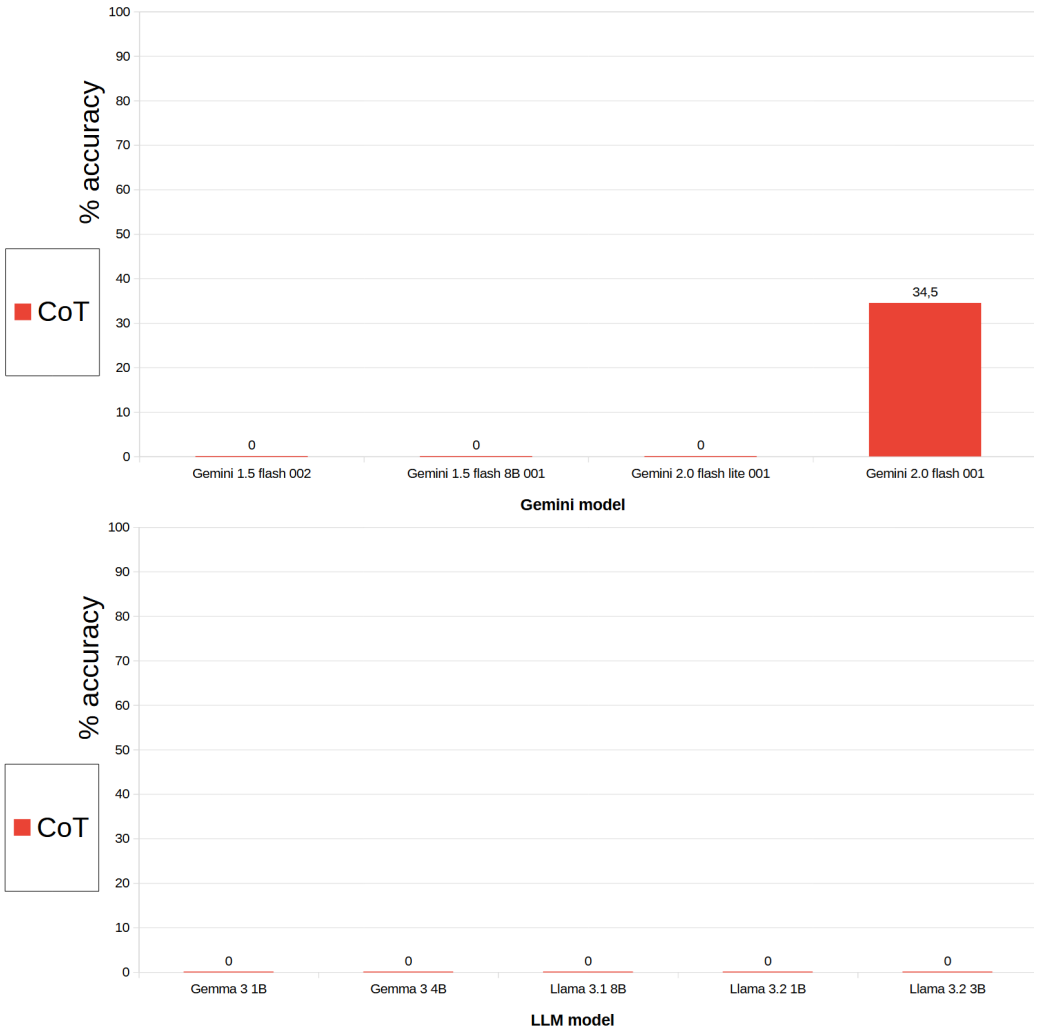


Figure 43: Impact of prompting strategies on cumulative accuracy by large language model on question 18. Graph's data are account-level aggregation.

Solve this problem that I just invented and called "poles and cylinders": there are three poles called stack1, stack2 and stack3. Stack2 and stack3 are empty while in stack1 are stacked three cylinders on the bottom The largest one called L, on top of this one the medium called M and on the top of the stack the smallest called S. The goal is to stack in the same order all the cylinders. In order to move cylinders, certain rules must be followed: cylinders can only be stacked in order of width from largest to smallest so S or M can be on top of L, only S can be on top of M, and no cylinder can be on top of S. In addition, cylinders can be moved from one pole to any of the other two, but only the cylinder from the top of a pole can be removed in order to move it, while those below are blocked as long as there is one on top. Solve the problem writing a maxium of 19 steps

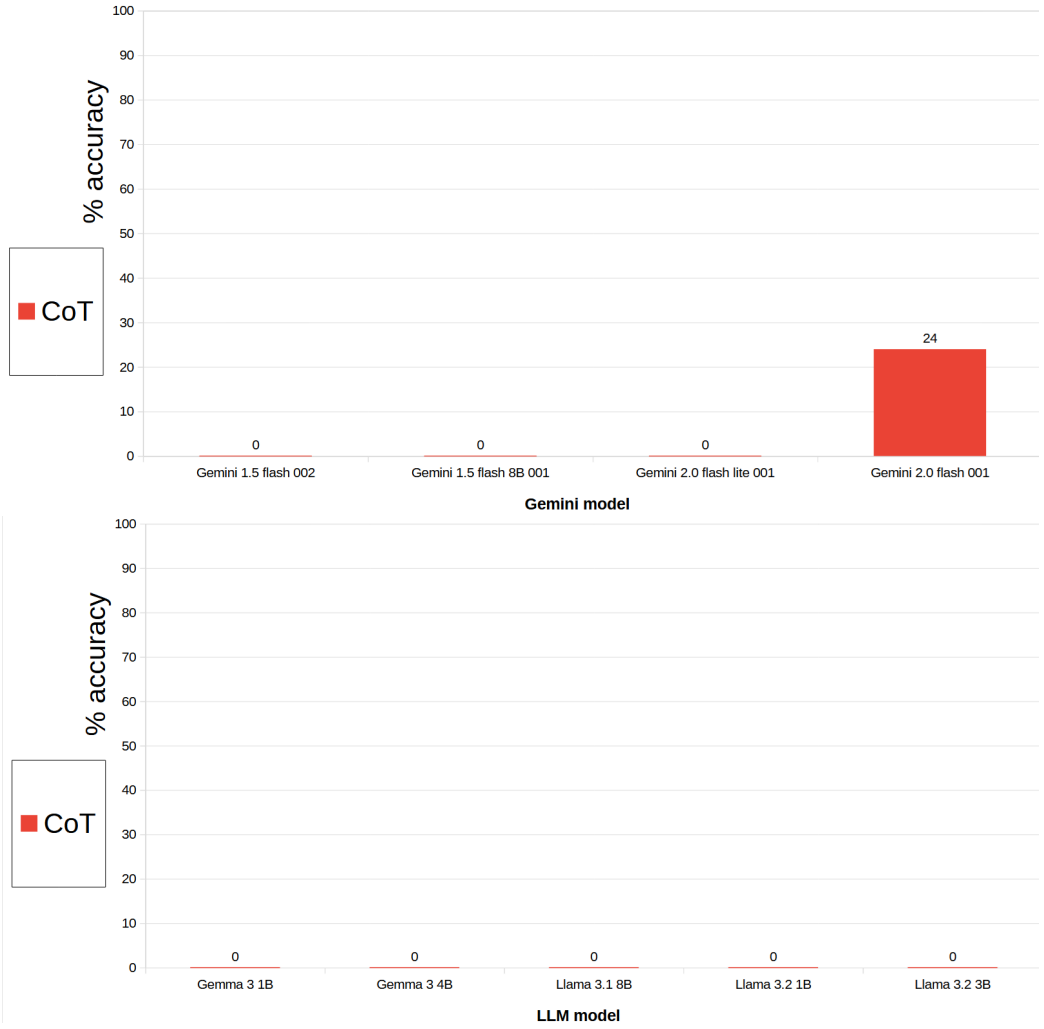


Figure 44: Impact of prompting strategies on cumulative accuracy by large language model on question 19. Graph's data are account-level aggregation.

Solve the tower of Hanoi with 4 disks, naming the disks, from largest to smallest: d1, d2, d3 and d4. Also name the source peg stack1, the auxiliary peg stack2 and the destination peg stack3.

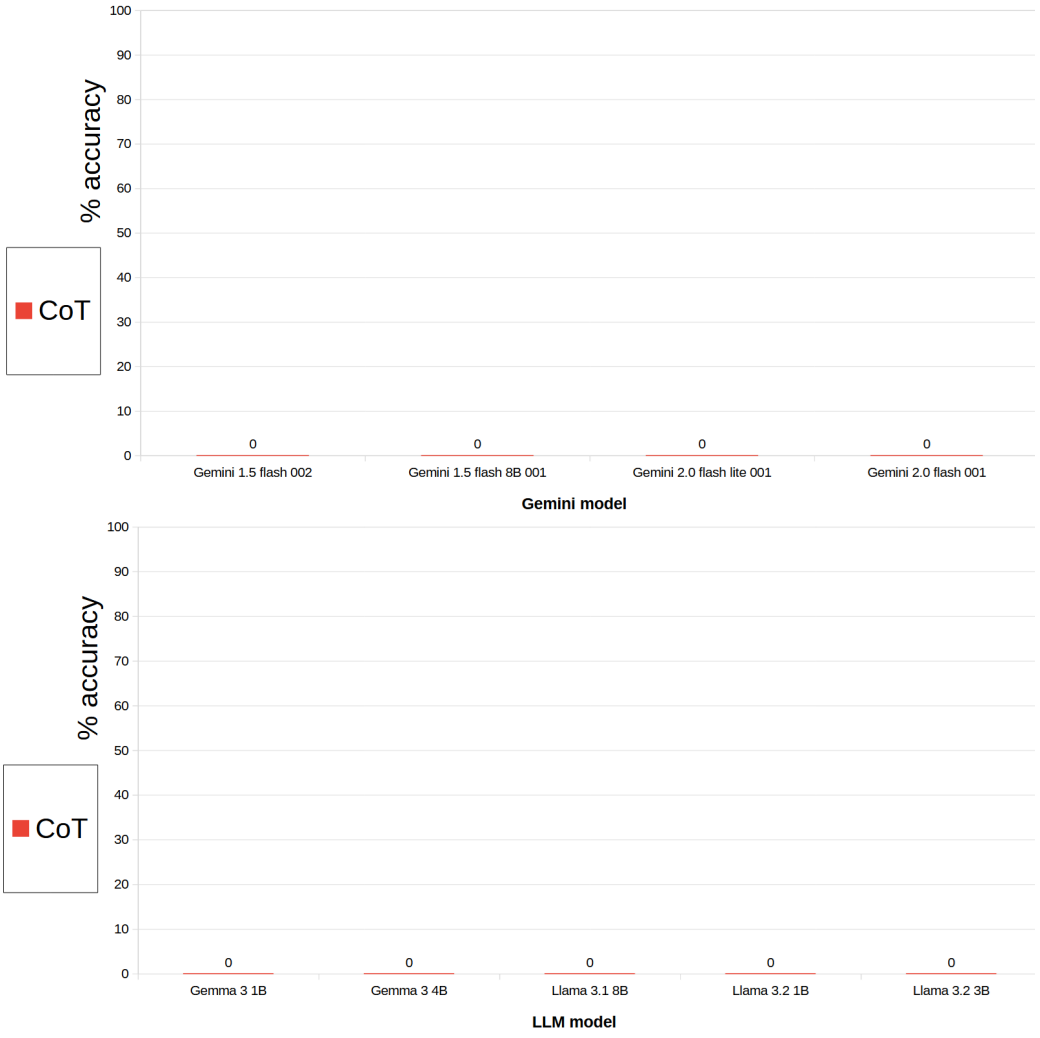


Figure 45: Impact of prompting strategies on cumulative accuracy by large language model on question 20. Graph's data are account-level aggregation.

Adding a cylinder and thus complicating the problem causes correctness to collapse, despite the simplification of considering the poles as stacks.

4.3.4 Ordered Stack

A slightly modified version of Hanoi Tower is now proposed. Accuracy is noted to be consistently close to zero.

Stacks are ordered queues whose main features in computer science are the two operations: pop and push. Push inserts an element at the end of the stack so that it is the last one, while pop removes the last inserted element. Example: by pushing an element B into a stack from the initial configuration [A] the result will be: [A,B]

There are three corresponding stacks:
Stack1: [A,B,C] (Any pop will remove C, which is the last element.
Stack2: []
Stack3: []

The problem must be solved by executing the pop and the corresponding push in the same step, since they must be considered as a single transaction.

The target configuration is [],[],[A,B,C].

There is an additional rule: elements must always be stacked alphabetically in each stack, so the valid configurations of a stack are only: [] or [A] or [A,B] or [A,C] or [A,B,C] (The order of the elements cannot be changed).

Solve the problem by writing a maximum of 19 steps

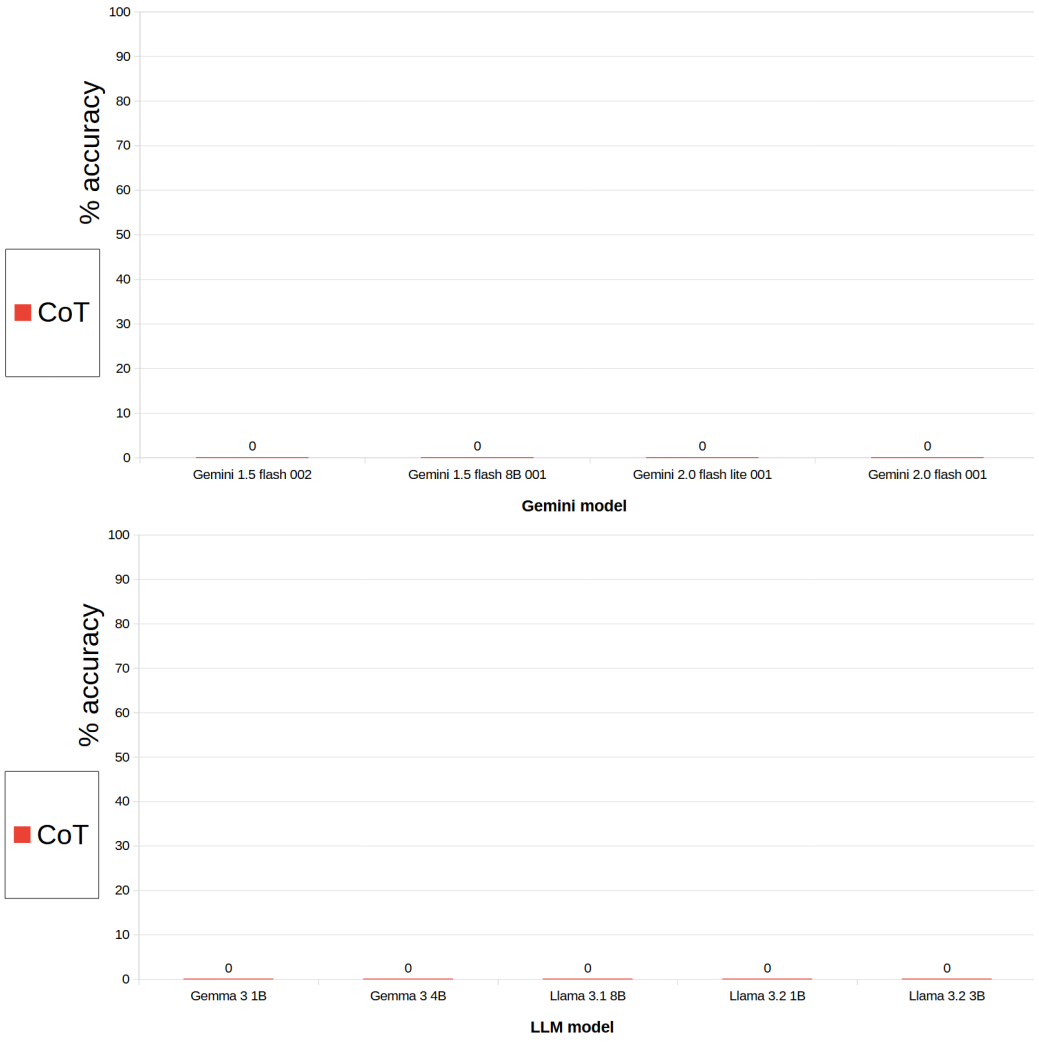


Figure 46: Impact of prompting strategies on cumulative accuracy by large language model on question 21. Graph's data are account-level aggregation.

There are 3 stacks whose graphic representation has the elements ordered from left to right: [first inserted, second inserted, third inserted, 4th inserted, ...], two are empty while one contains 4 elements:
Stack1: [B,A,D,C]
Stack2: []
Stack3: []
Using only pop and push operation coupled (execute both the operations in the same step) reach this final configuration:
Stack1: []
Stack2: []
Stack3: [A,B,C,D]
Solve the problem by writing a maximum of 19 steps

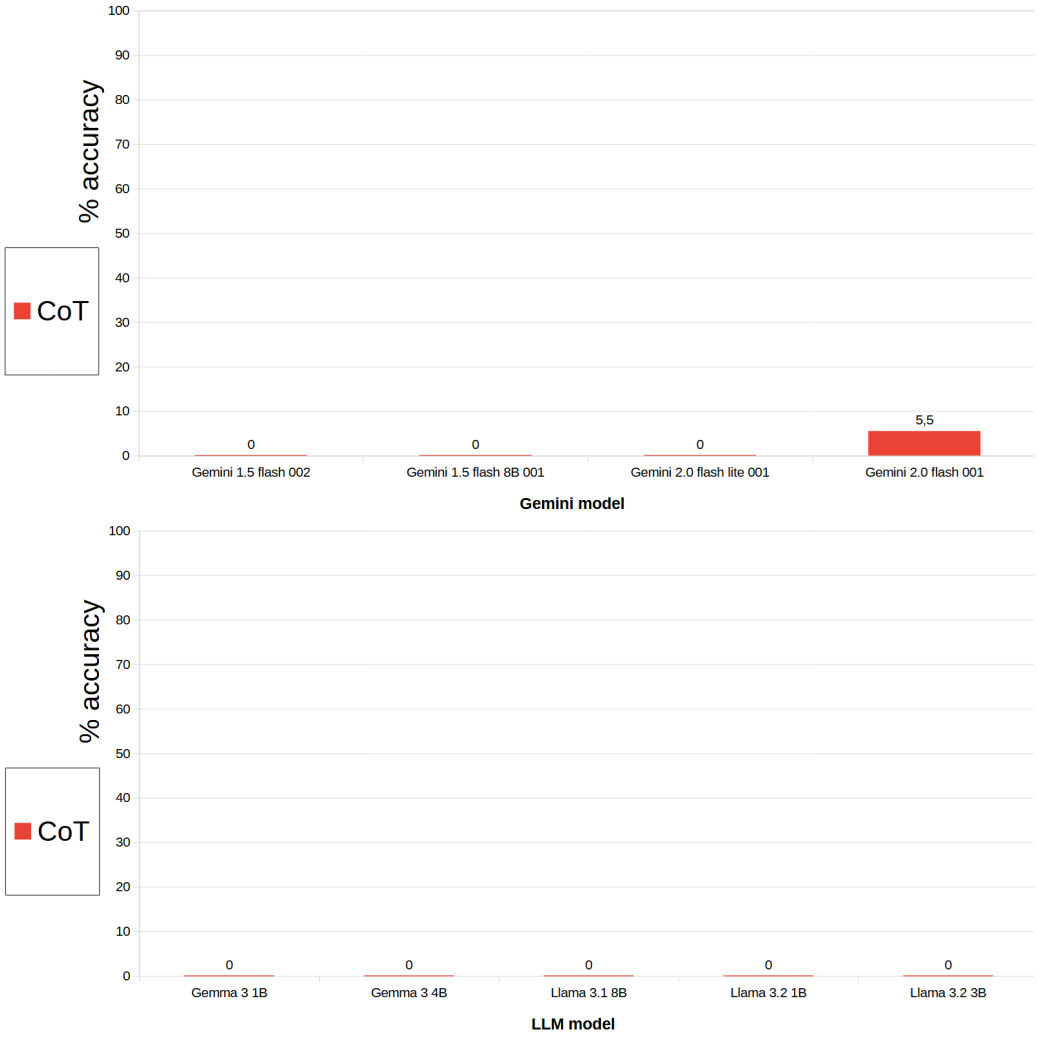


Figure 47: Impact of prompting strategies on cumulative accuracy by large language model on question 22. Graph's data are account-level aggregation.

4.4 Benchmark

By resizing the data to category-level aggregates, it is possible to obtain more general benchmarks which, when applied only to the most recent versions of the models, can show us the state of the art.

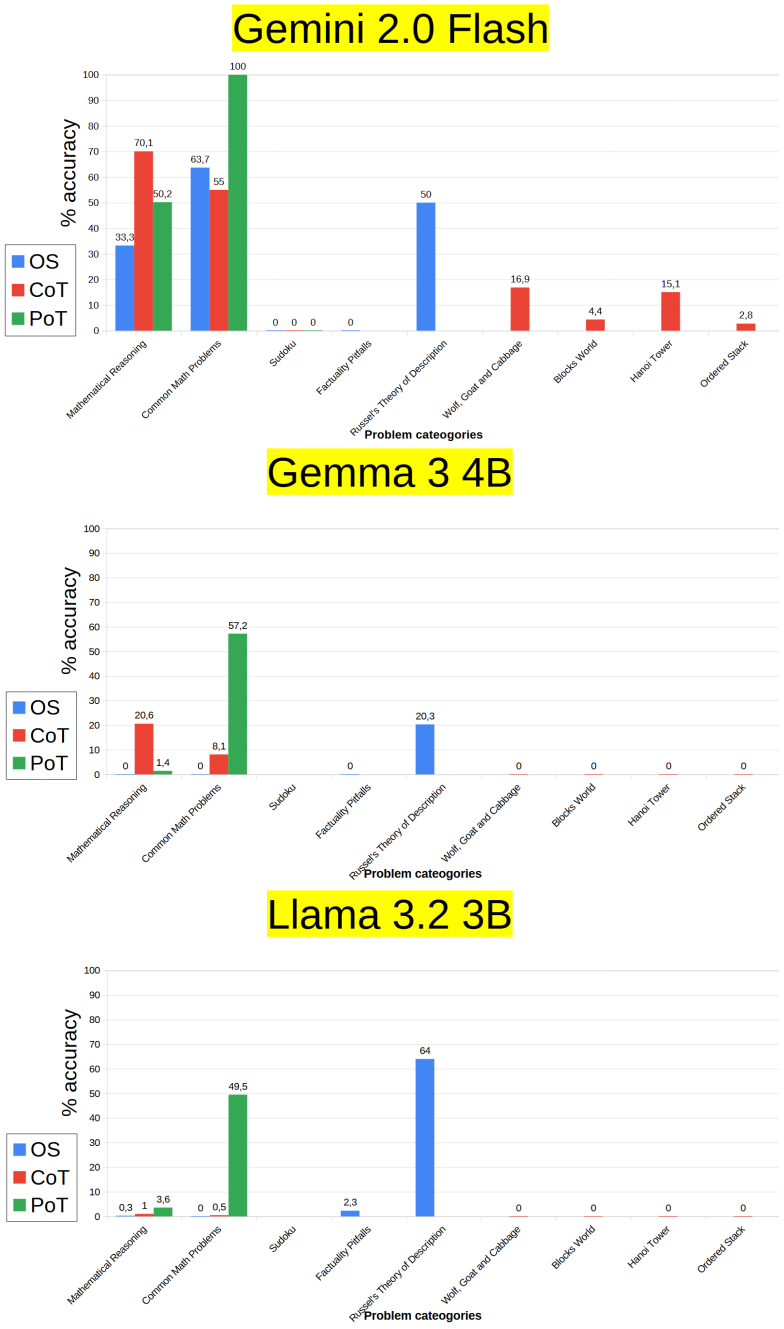


Figure 48: Impact of Prompting Strategies on Cumulative Accuracy by Problem Categories. Graph's data are category-level aggregation.

5 Conclusions

The results, analyzed by area, show that in **mathematical reasoning**, all tests show an improvement in Gemini's mathematical problem-solving skills as versions progress. Figure 27 is a clear example of a mathematical question for which Program of Thought (PoT) is inefficient and other prompting methods achieve excellent results. In this category, Gemma and Llama obtain inconclusive results, likely because the tests' difficulty is not adequate for the parameters of the models used.

In **common math problems**, Gemini achieved excellent results and the PoT methodology significantly increases the accuracy of the answers. Gemma and Llama show some difficulties in this category, however the use of PoT enabled the models to achieve promising results.

Sudoku was only tested on Gemini, and in all versions, the accuracy was 0%. The answers provided were always fairly consistent with the rules of Sudoku but completely invented and incorrect.

Some difficulties also emerge in **factual pitfalls**. In fact, especially with the one-shot (OS) methodology, the results are always close to being completely incorrect. Special mention should be made of figure 32, in which it appears that the zero-shot chain-of-thought (CoT) methodology allows reasoning that overcomes the pitfall. However, it should be noted that the data are highly inconsistent: accuracy decreases as Gemini versions progress and, even in open models, remains in the $3\% - 9\%$ range, except for Llama 3.2 3B, which reaches 33% with CoT. It should also be noted that the results in figures 32 and 33 may be overestimated since the correct answer is "0" and in some cases the agent's refusal to answer may have been considered a correct answer.

In the **Russell's theory of descriptions category**, despite the noticeable difficulties, the open models with more parameters and Gemini 2.0 flash show promising results. The latter agent even reaches the threshold of total correctness in one case. On the other hand, the Gemini 1.5 family and open models with few parameters always score 0%.

In **sequential problem solving**, the ability to manage overlapping reasoning plans in planning was tested. The results are quite clear. In fact, apart from figure 38, which shows a progressive improvement in accuracy in the Gemini models, reaching 54% for Gemini 2.0 flash thinking exp, in all other cases, the models hardly exceed the 2% threshold. In fact, in the figures following 38, Gemini 2.0 flash only significantly exceeds the threshold

in figures 43 and 44, with 35% and 24% respectively.

In conclusion can be inferred that LLMs have difficulty producing consistent answers when faced with problems for which they were not designed. It is therefore necessary to raise awareness of the areas of use in which their use is appropriate.

The study presented has considerable room for improvement:

- It analyzes only 3 of at least 7 goals of artificial intelligence.
- There are few questions proposed for each category.
- The LLMs analyzed are only those that do not require payment and can be hosted on low-cost hardware architecture.

Furthermore, the areas of use of LLMs are constantly increasing, often without studies to guarantee their safety.

References

[1] Ai risk management framework. URL: https://www.nist.gov/itl/ai-risk-management-framework. (accessed: 25.08.2025).

[2] Chloe Autio, Reva Schwartz, Jesse Dunietz, Shomik Jain, Martin Stanley, Elham Tabassi, Patrick Hall, and Kamie Roberts. Artificial intelligence risk management framework: generative artificial intelligence profile, 2024. URL: https://doi.org/10.6028/NIST.AI.600-1.

[3] Emily Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: can language models be too big? In pages 610–623, March 2021. DOI: 10.1145/3442188.3445922.

[4] Peter Norvig Blaise Agera y Arcas. Artificial general intelligence is already here. *Noema*, 2023. URL: https://www.noemamag.com/artificial-general-intelligence-is-already-here/.

[5] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: disentangling computation from reasoning for numerical reasoning tasks, 2023. arXiv: 2211.12588 [cs.CL]. URL: https://arxiv.org/abs/2211.12588.

[6] Noam Chomsky. Noam chomsky: the false promise of chatgpt. *The New York Times*, 2023. URL: https://www.nytimes.com/2023/03/08/opinion/noam-chomsky-chatgpt-ai.html.

[7] Benj Edwards. Openai reportedly nears breakthrough with reasoning ai, reveals progress framework. URL: https://arstechnica.com/information-technology/2024/07/openai-reportedly-nears-breakthrough-with-reasoning-ai-reveals-progress-framework/#gsc.tab=0. (accessed: 06.24.2025).

[8] Fish play pokemon. URL: https://www.youtube.com/watch?v=BEMJBHT5zBg. (accessed: 19.08.2025).

[9] Google. Embeddings. URL: https://ai.google.dev/gemini-api/docs/embeddings. (accessed: 06.24.2025).

[10] Google. Gemini 2.5: pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. arXiv: 2507.06261 [cs.CL]. URL: https://arxiv.org/abs/2507.06261.

[11] Google. Long context. URL: https://ai.google.dev/gemini-api/docs/long-context. (accessed: 06.24.2025).

[12] Google trends: agentic ai. URL: https://trends.google.com/trends/explore?q=%2Fg%2F11x2wlnqn_. (accessed: 07.08.2025).

[13] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmn, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier

Duchenne, Onur elebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vtor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland,

Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy,

Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.

[14] IBM. What are ai hallucinations? URL: https://www.ibm.com/think/topics/ai-hallucinations. (accessed: 06.24.2025).

[15] MS. SONAL BORDIA JAIN. Ethical considerations in artificial intelligence: navigating bias, fairness and accountability. *International Journal of Innovative Research in Science,Engineering and Technology*, 4(3), 2015. URL: https://www.ijirset.com/upload/2015/march/138_Ethical.pdf.

[16] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hllermeier. A survey of reinforcement learning from human feedback, 2024. arXiv: 2312.14925 [cs.LG]. URL: https://arxiv.org/abs/2312.14925.

[17] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023. arXiv: 2205.11916 [cs.CL]. URL: https://arxiv.org/abs/2205.11916.

[18] Rachel Metz. Openai scale ranks progress toward human-level problem solving, 2024. URL: https://www.bloomberg.com/news/articles/

2024-07-11/openai-sets-levels-to-track-progress-toward-superintelligent-ai.

[19] Joe Slater Michael Townsen Hicks James Humphries. Chatgpt is bullshit, 2024. URL: https://doi.org/10.1007/s10676-024-09775-5.

[20] Microsoft. Understand tokens. URL: https://learn.microsoft.com/en-us/dotnet/ai/conceptual/understanding-tokens. (accessed: 06.24.2025).

[21] OpenAI. Gpt-5 system card, 2025. URL: https://openai.com/index/gpt-5-system-card/.

[22] OpenAI. Openai charter. URL: https://openai.com/charter/. (accessed: 06.24.2025).

[23] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simn Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, ukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan,

Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, ukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mly, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cern Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William

Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. arXiv: 2303.08774 [cs.CL]. URL: https://arxiv.org/abs/2303.08774.

[24] Gerhard Paa and Sven Giesselbach. Foundation models for natural language processing – pre-trained language models integrating media, 2023. arXiv: 2302.08575 [cs.CL]. URL: https://arxiv.org/abs/2302.08575.

[25] Pokemon fire red - full game walkthrough. URL: https://www.youtube.com/watch?v=1MOjNA7I98g. (accessed: 19.08.2025).

[26] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: the sparsely-gated mixture-of-experts layer, 2017. arXiv: 1701.06538 [cs.LG]. URL: https://arxiv.org/abs/1701.06538.

[27] Stanford. The 2025 ai index report. URL: https://hai.stanford.edu/ai-index. (accessed: 06.24.2025).

[28] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition edition, 2018. ISBN: 78-0-262-03924-6.

[29] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ram, Morgane Rivire, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gal Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, Andrs Gyrgy, Andr Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle

Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Pluciska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Pder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Lonard Hussenot. Gemma 3 technical report, 2025. arXiv: 2503.19786 [cs.CL]. URL: https://arxiv.org/abs/2503.19786.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[31] What is agentic ai? URL: https://www.redhat.com/en/topics/ai/what-is-agentic-ai. (accessed: 07.08.2025).

[32] Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, Fazl Barez, Rahul Gupta, Jwala Dhamala, Jacob Merizian, Mario Giulianelli, Harry Coppock, Cozmin Ududec, Jasjeet Sekhon, Jacob Steinhardt, Antony Kellermann, Sarah Schwettmann, Matei Zaharia, Ion Stoica, Percy Liang, and Daniel Kang. Establishing best practices for building rigorous agentic benchmarks, 2025. arXiv: 2507.02825 [cs.AI]. URL: https://arxiv.org/abs/2507.02825.