Can AIs understand our world? Functionally grounding LLMs in interactive environments.

Clément Romac, Pierre-Yves Oudeyer, Thomas Carta

Developmental Systems

Flower's team blog: developmental AI and cognitive sciences

Can AIs understand our world? Functionally grounding LLMs in interactive environments.

Clément Romac, *Thomas Carta*, *Pierre-Yves Oudeyer*

Feb 6, 2025

18 minute read

ChatGPT, GPT-4, Mistral, Gemini… In recent months, a flurry of “language models,” mostly developed by major companies such as Google, Facebook, and OpenAI, have emerged. These intriguing tools, now widely used by the general public (who hasn’t heard of ChatGPT at a family gathering?), all operate on the same principle: software capable of responding to text-based messages from users.

Whether it is answering questions about historical facts, telling jokes, or generating cover letters, these tools initially appear impressive in their capabilities. However, upon closer inspection, certain limitations become evident. Notably, these systems sometimes produce false or entirely fabricated information. This makes distinguishing truth from falsehood particularly challenging since the model itself cannot discern the difference and presents all outputs with equal confidence.

Some inaccuracies, however, immediately raise suspicion, especially when they seem completely disconnected from the world as we know it. Consider, for instance, this absurd explanation from ChatGPT: an anaconda cannot “fit, size-wise, in a shopping mall” because its length would pose a problem given the building’s ceiling height.

Despite their impressive capabilities, LLMs still fail at simple tasks by providing answers that appear disconnected from our world (such as the anaconda not fitting a shopping mall). Illustration generated with FLUX.1[dev]

Why do models miss the obvious? How can something so apparent to us escape a language model entirely? To understand this, we must delve into the nature of these

systems. Language models are based on a mathematical function with an enormous number of parameters—over 100 billion in today's models. These parameters are adjusted to produce the desired output for a given input. Using mathematical methods known as Machine Learning, these parameters are automatically tuned to match inputs and outputs in a training dataset (a set of input-output pairs). In practical terms, each input word is assigned a representation vector (a point in a multidimensional real-number space), and the output is a probability distribution over possible next words (within a predefined vocabulary). Training a language model involves exposing it to vast amounts of text and tuning its parameters so that it assigns the highest probability to the correct next word in each context. To generate text, the model starts with an initial phrase, selects the most probable next word, and iterates this process, appending each newly chosen word to the input.

This training principle predates the recent explosion of language models. Although employed for decades, earlier iterations produced encouraging but far less impressive results compared to today. What changed? First, the mathematical function that must be tuned evolved. Until 2017, language models primarily relied on recurrent neural networks (RNNs), which process one word at a time and use a memory matrix to retain information across words. In 2017, a new model called the transformer was introduced. Unlike RNNs, transformers consider all words in the context simultaneously, using a mechanism called attention to relate representation vectors of different words. Moreover, transformers can perform many operations in parallel, enabling the training of models with tens of billions of parameters and vast datasets.

Because language models produce a probability distribution for the next word, they can generate text but also calculate the likelihood of one phrase following another. For instance, if GPT-3 is asked about its favorite color, the next word probabilities reflect the preferences expressed by humans during training.

The symbol grounding problem

This passive learning gives models some knowledge about our world. However, this knowledge is purely "bookish". It

doesn't stem from direct experience of reality. Yet, it is this direct experience that enables us to intuitively understand whether, say, an anaconda can navigate the aisles of a supermarket.

How can AI acquire this capability? And, more fundamentally, is this even desirable? Beyond attempting to mimic natural processes, grounding words in the real world is crucial if we want models to be genuinely useful beyond generating jokes, summarizing texts, or structuring documents. For example, we need to share a common foundation of knowledge with these models so that the words we use to communicate with them carry the same meaning or refer to the same concepts.

Take the example of asking a robot to "pick up an apple." The robot must connect the word "apple" to the physical reality of an apple, accounting for variations in its color, size, shape, and taste.

> *Language models extract factual knowledge from their training data (e.g., history books, recipes) and understand object relationships (e.g., knives cut, spoons don't). This allows them to predict common interactions, forming a limited model of the world. Humans learn similarly, especially children, with stories aiding understanding. However, human learning also involves interaction within a sociocultural context, where words are tied to physical and social realities. Unlike humans, language models lack this grounding in the real world.*

This issue of grounding is well-known in research. It was formalized in 1990 as the symbol grounding problem by cognitive scientist Steven Harnad, who posed the following question: Can a system of rules defining possible combinations of symbols (such as words or hieroglyphs) inherently contain meaning and refer to objects or concepts outside this system? Intuitively, the symbol grounding problem examines how symbols—such as the words we use—can be associated with the world around us to carry meaning when interpreted in context. Psychologists and linguists have extensively studied grounding, and it has also inspired research in Machine Learning. Much of this research has focused on associating object names with their visual representations. This has led to numerous image generators capable of producing visuals from textual descriptions. Some of these advancements have been applied to modern language models. For example, models like

GPT-4 can now accept both text and images as inputs. Yet, the broader grounding of these models in the physical world remains unsolved. The current mechanisms still fail to capture the meaning of certain concepts, such as physical properties, as evidenced by ChatGPT's anaconda scenario.

What would it take for a language model to grasp the fundamental concept of gravity, a cornerstone of our reality? Could such grounding enable a robot, conversing like ChatGPT, to perform practical tasks in the physical world?

Functional grounding implies active learning mechanisms: one must interact with its environment through actions and perception to functionally ground symbols. Illustration generated with FLUX.1[dev].

Answering these questions requires exploring another type of grounding: functional grounding. This approach focuses on the symbols used to act in a given environment and predict the outcomes of actions, enabling problem-solving within that environment. By solving problems, the model can ground the physical dynamics or rules of its environment.

In July 2023, we introduced "Grounding Large Language Models in Interactive

Environments with Online Reinforcement Learning" (aka GLAM). In this paper, we proposed the first approach to functionally ground LLMs.

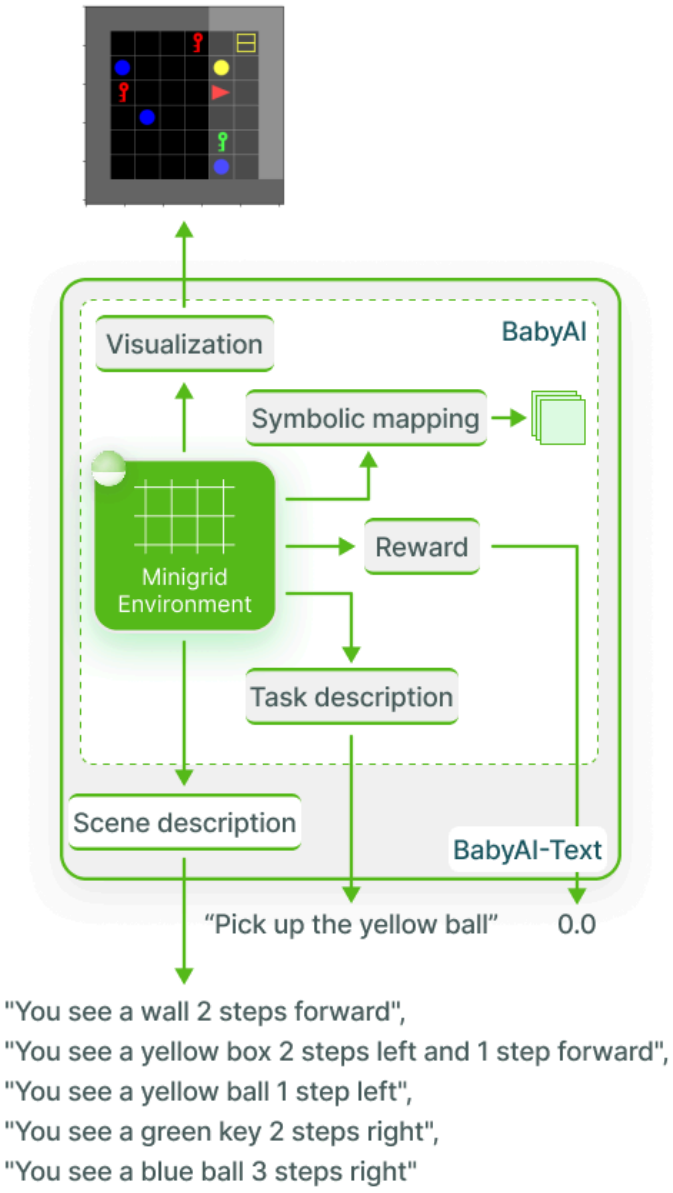
Towards functional grounding

This way of grounding reality in our representations, particularly the actions that change our world, is something we naturally rely on, especially when using our inner speech (endophasia) to list actions needed to achieve a goal. For instance, when cooking, it is common to mentally plan the next steps to create the desired dish. Symbols (such as actions like "cut," "cook," etc.) are used to act on our environment and to predict its state after the interaction. The grounding here is specific to each environment encountered —"advance" for example, does not mean exactly the same in the context of a chess game and in our everyday world.

How can this mechanism be applied to an AI system? How can we link a language model to an environment, whether it is the physical world or a computer simulation, in a way that connects symbols, actions, and the outcomes of these actions? For this initial study of functional grounding, we needed to create a dedicated environment that would allow us to isolate functional grounding from other forms of grounding and study its unique properties.

> *The concept of an agent is fundamental and highly prevalent in the field of Artificial Intelligence. We adopt the definition provided by Stuart Russell and Peter Norvig: an agent is "any entity that can be considered as perceiving its environment through sensors and acting upon that environment via effectors".*

We required an environment in which an agent could act and where its actions would affect that environment. Moreover, the agent needed to decide which action to take based on the outcomes of its previous actions. In practice, our agent is nothing other than a language model, which serves as both the body (capable of action) and the brain (capable of decision-making).



We introduce BabyAI-Text as a testbed for functional grounding. It extends the classic BabyAI test bed initially designed for other forms of grounding.

BabyAI-Text, an experimental testbed for functional grounding

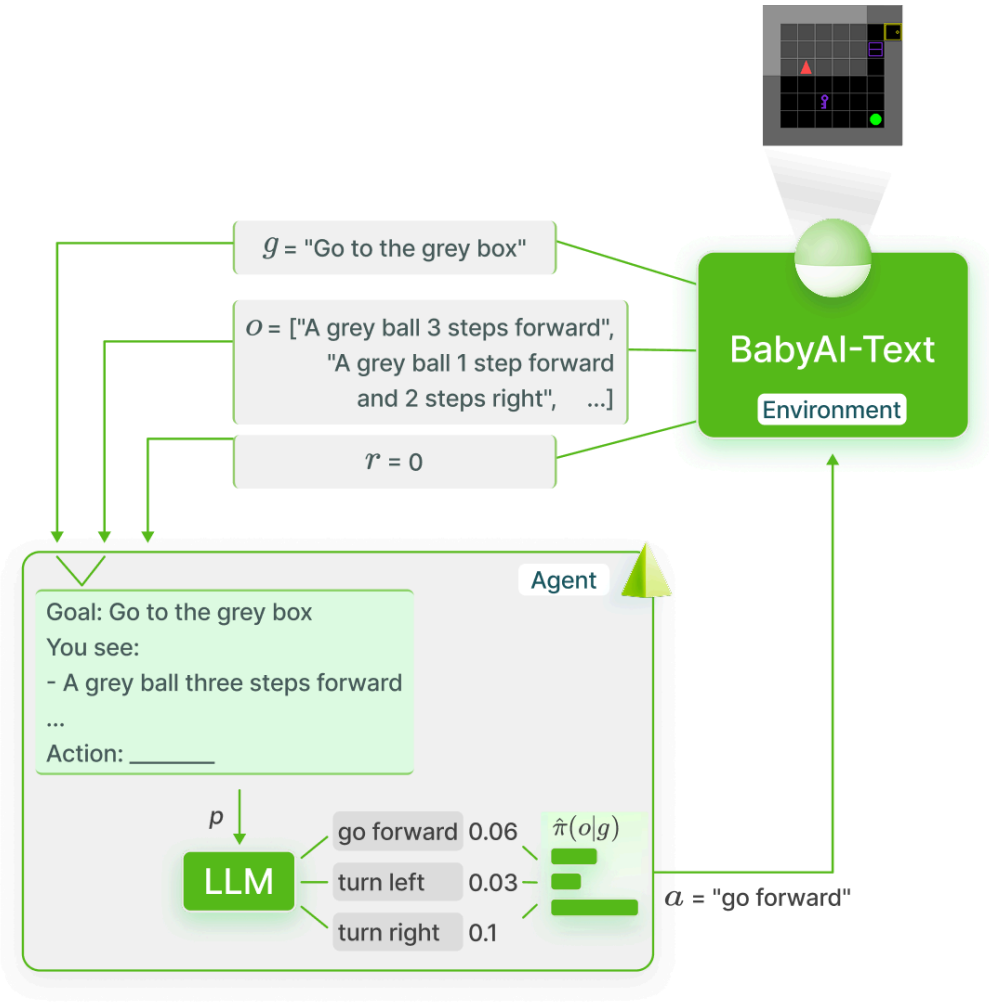
We proposed to use an interactive textual game called *BabyAI-Text. While more complex environments, such as those involving visual perception (e.g., a robot with a camera interacting with objects in a kitchen), are enticing, they involve additional forms of grounding (like linking an object's name to its various visual facets). Instead, we opted for a simulation simple enough for its elements to be directly described in text, yet rich enough to explore different properties of functional grounding. Our system resembles early video games, such as [Zork], a turn-based text adventure where a character or agent receives a description of what it "sees" and can send*

text-based commands (e.g., "attack the dragon with the sword") back to the environment. When the game begins, the player is given a goal (e.g., "defeat the dungeon monster") along with an initial description of their surroundings. Achieving the goal requires performing various actions, more or fewer depending on the chosen strategy. Each turn, the player types the desired action the agent must perform and receives an updated description of their perception.

In BabyAI-Text, the agent operates in a room where it can move using three commands: "move forward," "turn right," and "turn left." The room contains objects the agent can pick up and move. The agent receives a goal such as "Place the red ball next to the blue box." On each turn, it can only see part of the room and must choose an action from six possible options (three for movement and three for object interaction). Once the goal is reached or the turn limit is reached, the room's content is reset, and a new, randomly assigned goal (possibly composed of multiple simpler goals) is given.

GLAM, functionally grounding through online interactions

In our experiments, the agent is controlled by the large language model [Flan-T5] (with 780 million parameters, developed by Google in 2022). The "brain" of the agent was therefore initially trained by its creators to predict the right next token given text inputs drawn from millions of documents. It must now learn to judiciously use the vocabulary associated with its interactions in BabyAI-Text. For example, when the model chooses to use the command "move forward," it must associate the word with what moving forward entails in the environment (e.g., the perception changes, objects in front of the agent move closer, and if there is an object already in front, it remains in place). We call the method enabling this learning **GLAM** (Grounded LAnguage Model).



In GLAM, we assemble the textual perception given by BabyAI-Text into a prompt that is given to the LLM to functionally ground. This LLM then selects an action to perform by compyting the probability of each possible action in BabyAI-Text.

To perform functional grounding, instead of direct agent-environment interaction, we could have opted to show the language model explanations (in sentence form) of the environment's dynamics and train it to reproduce those explanations (i.e., predict the next word in the provided explanation). However, it is not always straightforward to encode the physics of an environment in sentences. For instance, explaining that movement in an environment changes what is subsequently perceived is challenging. This concept seems easier to grasp through direct interaction with the environment and learning from the results of those interactions. Furthermore, learning to recite explanations does not necessarily mean the model has anchored the dynamics or can use them to solve problems in the environment. There is no guarantee.

Another possibility would be to provide the model with input-output pairs, where the input contains the goal to solve and the agent's observation, and the output specifies the chosen action. But where would these examples come from? Should we ask a human to interact with the environment for thousands of turns? Or create another artificial agent to explore the environment and provide the language model with these examples? A major drawback of this method is that passively inferring causal effects from observing another agent's actions can introduce confounding factors—information that influences both the agent's decisions and the results (the subsequent observations). For instance, an autonomous car learning to drive by observing humans might infer that braking makes pedestrians appear, as it often observes pedestrians wanting to cross when the car brakes, potentially leading the car to avoid braking altogether.

This is why, in our study of functional grounding, the proposed agent generates (or rather assembles in a predefined way, via a "prompt") a text containing the goal, the observation description, the list of possible actions, and a question asking what action to take. This text serves as input for the language model, which is expected to provide the continuation: the next action.

It is essential to remember that a language model is a mathematical function used to compute the probability of a complete sentence following a given text. This principle is central to GLAM. Instead of generating the most likely text sequence following the input (which might not correspond to an action possible in the environment), GLAM uses the language model to directly calculate the probability of each action given the input. GLAM evaluates all options ("move forward," "turn right," etc.) appended to the prompt ending with the question and selects the action with the highest probability according to the language model.

The agent, equipped with the prompt assembling information returned by the textual environment, effectively has both a brain (the language model) and a body. How does it update its brain's knowledge based on its body's actions? In the GLAM approach, we use Reinforcement Learning. In this subfield of Machine Learning, an agent tests various strategies (associating actions with observations) to maximize a reward provided by the

environment after a series of interactions. The strategy parameters are gradually adjusted without experimenter intervention to achieve the best possible reward. This method enabled DeepMind to develop [AlphaGo], the Go-playing agent that defeated the world champion, and OpenAI to train an agent to solve a Rubik's Cube using a robotic hand [2].

Is this enough to functionally ground a language model on its environment? We conducted a total of 1.5 million interactions between Flan-T5 and BabyAI-Text to train the agent to solve various types of goals in different rooms. Tested on 1,000 new room-goal combinations unseen during training, the grounded version of Flan-T5 (named GFlan-T5) successfully completed 89% of tasks requiring it to reach or retrieve an object, compared to only 11% for Flan-T5 without any grounding. Remarkably, GFlan-T5 maintained an 87% success rate even when the objects in the room were entirely new (training tasks only involved balls, boxes, and keys, whereas tests included chairs, tables, and cars). When these new objects were assigned invented names ("axfe," "xolo," "dax"), GFlan-T5 still succeeded in 88% of tasks. These results demonstrate how GLAM's functional grounding improved Flan-T5's handling of vocabulary tied to environmental dynamics (e.g., "move forward," "turn") without disrupting unrelated vocabulary, such as object names.

Finally, we compared GLAM's active grounding approach to a passive grounding method. We created an expert agent specifically for BabyAI-Text with access to more information than the agent presented above. It could solve all tasks using a hand-designed, optimal strategy (which does not generalize to other environments). Using this expert, we recorded as many "turns" as GFlan-T5 explored. Each turn's goal, observation, and possible actions were formatted as a prompt, and the same Flan-T5 model was trained to predict the expert's chosen actions (instead of discovering on its own a strategy using GLAM). This created a passive functional grounding, where the language model learns to choose the correct action without interacting with the environment.

Results showed that passive learning was less effective than GLAM's active approach, even when provided with examples of the best actions for each turn. Moreover, when room objects changed, the passively trained agent performed significantly worse. This gap is explained by the limited exposure of passive grounding to diverse examples. For instance,

GFlan-T5 learns to correct errors (e.g., retracing steps), while passive agents lack such adaptability. Additionally, passive grounding may introduce confounding factors that are challenging for the model to untangle [3]. Lastly, the active Reinforcement Learning used by GLAM trains the model to select actions aimed at achieving the final goal, emphasizing long-term strategy acquisition.

Towards grounded LLMs in robots?

With the introduction of a novel approach to the famous symbol grounding problem, as well as a pioneering method for functionally grounding a language model, a new path opens up toward future AI systems more deeply embedded in our world. This is, in fact, only an initial step, currently tested on relatively small models in a limited framework. However, this work could inspire other research teams to better understand and address the functional grounding of larger language models.

Another important question is how such grounding can be implemented in our world. Recent research [4][5] has integrated language models into robots interacting with the physical world, using various Machine Learning models to describe what the robot's camera perceives. These studies have demonstrated that, similar to the work with GLAM, the language models can select actions to solve tasks assigned to the robot. However, they do not involve functional grounding to align the language model with the environment or to correct its decision-making in case of errors. Such a framework would require studying not only the functional grounding involved in decision-making but also the grounding of the mechanisms that transform camera images into textual descriptions and the grounding of the process that converts language model-described actions into sequences of electrical impulses sent to the robot's motors.

Finally, what about the impact functional grounding has on the language model when it is later used to generate text and answer questions? Is this the best way to imbue its responses with common sense? These are also open questions this first study of functional grounding raises.. For instance, would functionally grounding a language model like the one powering ChatGPT in an environment where it must place objects of the correct size into a

given container suffice to ensure the conversational agent no longer makes mistakes like the “anaconda in the shopping mall?

Cite GLAM

```
@InProceedings{pmlr-v202-carta23a,
  title = {Grounding Large Language Models in Interactive Environments with Online Reinforcement Learning},
  author = {Carta, Thomas and Romac, Cl\'{e}ment and Wolf, Thomas and Lamprier, Sylvain and Sigaud, Olivier and Oudeyer, Pierre-Yves
  booktitle = {Proceedings of the 40th International Conference on Machine Learning},
  pages = {3676--3713},
  year = {2023},
  editor = {Krause, Andreas and Brunskill, Emma and Cho, Kyunghyun and Engelhardt, Barbara and Sabato, Sivan and Scarlett, Jonathan
  volume = {202},
  series = {Proceedings of Machine Learning Research},
  month = {23--29 Jul},
  publisher = {PMLR},
  pdf = {https://proceedings.mlr.press/v202/carta23a/carta23a.pdf},
  url = {https://proceedings.mlr.press/v202/carta23a.html},
}
```

References

- Carta, T. et al. (2023). Grounding Large Language Models in Interactive Environments with Online Reinforcement Learning. Proceedings of the 40th International Conference on Machine Learning
- Bruner, J. (1990). Acts of meaning. Harvard University Press.
- OpenAI et al. (2019). Solving Rubik's Cube with a Robot Hand.
- Gasse, M., Grasset, D., Gaudron, G., & Oudeyer, P. (2023). Using Confounded Data in Latent Model-Based Reinforcement Learning. Trans. Mach. Learn. Res., 2023.
- Ahn, Michael et al. (2022). Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. Conference on Robot Learning.
- Huang, Wenlong et al. (2022). Inner Monologue: Embodied Reasoning through Planning with Language Models.

Share this post!

-->