COGNITIVE TRAINING FOR LANGUAGE MODELS: TOWARDS GENERAL CAPABILITIES VIA CROSS-ENTROPY GAMES

CLÉMENT HONGLER[1,2] FRANCK GABRIEL[3,1] VALENTIN HARTMANN[1] ARTHUR RENARD[1] ANDREW EMIL[1]
[1]XENT LABS [2]EPFL [3]UNIVERSITÉ LYON 1

Abstract. Defining a constructive process to build general capabilities for language models in an automatic manner is considered an open problem in artificial intelligence. Towards this, we consider the problem of building a curriculum of tasks that grows a model's general capabilities. We argue that any solution to this problem should lead to that of an priori simpler problem, which we call (automatic) *relevant skill discovery.*

We provide a concrete framework for this task, using a family of tasks called cross-entropy games, which we postulate is universal in a suitable sense. We show that if it is possible to grow the curriculum for relevant skill discovery by iterating a greedy optimization algorithm, then, under natural assumptions, there is essentially only one *meta-objective* possible (up to a few hyperparameters). We call the resulting process *cognitive training.*

We postulate that, given sufficiently capable language models as players and meta-samplers and sufficient training time, cognitive training provides a principled way to relevant skill discovery; and hence to the extent general capabilities are achievable via curriculum learning, then cognitive training would be a solution.

1. Artificial General Intelligence and Cognitive Training

The last decades saw spectacular progress on several fronts for AI. In particular we saw:

- Models that could generalize beyond their training dataset.
- Reinforcement learning that could reach super-human performance on specific tasks.
- Pre-trained Large Language Models (LLMs) that could learn to perform certain tasks with in-context learning, and numerous more specific tasks, given a suitable training environment.

In spite of these achievements, there is a growing consensus that a number of ideas are lacking to go towards "true Artificial General Intelligence (AGI)". In this paper, we start with a short summary of what we expect from AGI starting from seminal works [Tur50, LeHu07] and propose to study an (apparently simpler) problem, which is that of (open-ended) relevant skill discovery.

To study this problem, we propose a training framework (based on an elaboration of the framework introduced in [HoEm25]) which we call *cognitive training*: the idea is to grow a curriculum of games in a specific space of tasks, called *cross-entropy (xent) games*, endowed with a suitable notion of *transfer value*: how learning one game teaches one to play another. We postulate that the learning achieved by any curriculum of tasks can be approximated by a curriculum of xent games.

We consider the problem of growing a curriculum of xent games in a greedy manner, by optimizing a meta-objective $\mathcal{O}$. We show that (perhaps surprisingly), given a few natural assumptions, a consistent meta-objective must take a very constrained form: from these, we derive an explicit formula for $\mathcal{O}$, that balances sparsity (inverse code length), quality (internal consistency), diversity (novelty), and external relevance (benchmark performance).

This leads us to *cognitive training.* The idea is to start with one (or more) base auto-regressive model $\mathcal{M}$, find a suitable space of xent games $\mathcal{G}$. At every step of the curriculum, a meta-sampler $\mathcal{M}_{\mathcal{M}}$ then generates games on $\mathcal{G}$ to optimize $\mathcal{O}$, to grow a curriculum that brings relevant skill discovery to $\mathcal{M}$.

Our reasoning thus leads (from a number of assumptions) to the following idea: *if it is possible to greedily generate a curriculum of tasks bringing general capabilities to a model, then it is possible to replicate this with a greedily built curriculum of xent games, and then the meta-objective formula must take a very specific form.*

1.1. **Definition of Artificial General Intelligence.** The first notable attempt at a measure of artificial intelligence is probably Turing's imitation game [Tur50], leading to the famous Turing test. Across decades, various definitions of intelligence in the context of AI were provided, leading to a notable and influential synthesis in [LeHu07]:

Intelligence measures an agent's ability to achieve goals in a wide range of environments.

S. Legg and M. Hutter

Date: March 24th, 2026.

A framework built at the intersection of Occam's Razor (in the form of Algorithmic Probability) and of Reinforcement Learning [LeHu06] highlighted notably a theoretically well-posed, but uncomputable, approach to the so-called Universal AGI problem: the AIXI paradigm [Hut05], blending a universal prior on world models (based on Kolmogorov's complexity) and objective maximization.

Arguably, the part that remains a little vague is associated with the "wide range" in the above definition: is there a good measure of generality? Are humans general according to any such definition? Is there a process through which this generality is discovered?

Despite the impossibility to apply this framework directly (due to its uncomputability) and the lack of answers about generality, AiXi gives a direction for what we would expect theoretically from an AI going towards AGI: models should interact with an environment, perform tasks within them, and be rewarded. Two decades of progress in the field of Reinforcement Learning (RL) followed. In the context of language models, this was discussed for quite some time [MJB15]. However, as discussed in the next subsection, the progress has only recently accelerated thanks to the rise in prominence of pre-trained large language models.

1.2. **Large Language Models in Early 2026.** Pre-trained large language models (LLMs) have emerged in the last 5 years as one of the most powerful starting points to perform reinforcement learning (RL). Performance of fine-tuned language models trained on RL environments has demonstrated that, with a reasonably large amount of training episodes, such models could achieve spectacular performance on numerous tasks, including coding and mathematical problem solving.

These successes of RL and the unparalleled growth of AI datacenters have led to various debates and speculations about AGI timelines. In the current state of affairs, it is reasonably clear that we have the following:

- Pre-trained models can learn a large number of tasks if those can be formulated within a reinforcement learning environment with rewards of sufficient density.
- Pre-trained models still require a large number of training episodes compared to what a human would require to learn a task learnable within a given environment.
- The performance is uneven: the generalization from already seen tasks to unseen tasks is not impressive. For instance, agents trained on one programming language may struggle on other languages; models able to solve advanced math problems may struggle with some basic counting tasks.

As a result, in spite of their superhuman abilities on specific tasks, it is hard to trust LLMs to perform adequately on unseen problems, in particular when the number of allowed attempts is not large. In other words: pre-training has allowed one to reliably start doing RL on many specific given tasks, but in a way that does not reliably generalize outside of these tasks. This echoes a bit the state of affair of supervised learning before the deep learning revolution, where models could fit training points, but could not generalize well outside of those.

In the next subsection, we outline a new framework based on games to address this problem, which we call *cognitive training.*

1.3. **Cognitive Training: Goals.** As discussed above, the current problem with LLMs is not that they cannot learn tasks given a suitable RL environment, but that they have a hard time performing (or quickly learning to perform) tasks for which they have not been trained already. Given a new task, constructing a specifically relevant RL environment and post-training a model on it can be very difficult or impossible.

The question that we aim to address can be understood as building extended pre-training using *games* (environments with an emphasis on reward choices):

Problem 1. Can we find a curriculum of compact games $(G_k)_{k\geq 0}$ such that an LLM $\mathcal{M}$ trained on it will be maximally ready for new/unseen games given some data, architecture, and compute constraints?

Mathematically, the above problem seems to assume a prior on new games, and is hence somewhat ill-posed (we have no way to set this prior). Like for supervised learning (where e.g. computer vision problems could be solved without a clear prior on the set of images corresponding to a given label), it is reasonable to expect that one can achieve good performance with the right methods, without expliciting (or sampling from) such a prior. By Occam's razor, if we find a curriculum of compact games (at finite k) that yields improved performance on an external metric E (e.g. an aggregation of benchmarks), then it is reasonable to expect that this curriculum will be *generally* valuable: a model trained on such games is likely (in an informal sense) to generalize well to tasks beyond the specific tasks of E.

Pre-training is in fact the most emblematic example of a compact training game: predicting the next token, a most universal and abstract task, leads us quite far in terms of general capabilities. Given a dataset, this is, however, a game that can only be played once (and with diminishing returns) leading to the question: what are, besides pre-training, the most useful training games?

Interestingly, we can perform a substantial reduction to the above question by asking the following, which seems a much simpler problem a priori:

Problem 2. Can we find a curriculum of compact games $(G_k)_{k\geq 0}$ such that an LLM $\mathcal{M}$ will keep discovering new skills, while maintaining existing capabilities?

In the context of a game space endowed with an appropriate structure of transfer value (see Definition16), this question is much easier to pose properly. The following seems intuitive:

Claim 3. Any convincing solution to Problem 1 should also yield a solution to Problem 2.

Now, the central (and perhaps surprising) claim of this paper is made in Sections 4.2–4.3:

Claim 4. If we can solve Problem 2 via a greedy method involving the optimization of a meta-objective $\mathcal{O}$, then from reasonable principles an explicit (and unique) formula for $\mathcal{O}$ can be found.

We call *cognitive training* the process of performing the meta-optimization for $\mathcal{O}$.

In the next subsection, we first lay out a framework to optimize cognitive training for LLMs, detailed in the rest of the paper.

1.4. **Cognitive Training: Implementation.**

The question raised in Problem 1 above suggests to formulate a meta-optimization problem: that of building a curriculum of games that turn out to be maximally useful to a (language) model $\mathcal{M}$ learning to play them (for a good definition of "usefulness"), on which one trains a model sequentially.

To implement this program, we need two ingredients:

- a space of games $\mathcal{G}$ on which the model $\mathcal{M}$ will be trained,
- a meta-objective $\mathcal{O}$ defined on games $\mathcal{G}$, used to select which games to add to the curriculum.

Given these two ingredients, $\mathcal{O}$ is optimized greedily, step by step, by a meta-sampler $\mathcal{M}_{\mathcal{M}}$, which outputs games (written in a specific language). The next game added to the curriculum to train model $\mathcal{M}$ is the one that maximizes the meta-objective $\mathcal{O}$. In the rest of this subsection, we briefly discuss principles for the design of $\mathcal{G}$ and $\mathcal{O}$; details are presented in Sections 3 and 4 respectively.

1.4.1. *Game Space Design Elements.*

The desiderata for the space of games $\mathcal{G}$ aimed at developing the cognitive capabilities of a model $\mathcal{M}$ are numerous:

- The games should be amenable to fast training.
- It should be easy to generate a wide diversity of games in $\mathcal{G}$ with fairly concise code.
- The space should allow for robust transfer exploration: if a certain curriculum (based on certain games not necessarily in $\mathcal{G}$) teaches $\mathcal{M}$ some skills, the same can be achieved by a curriculum of games in $\mathcal{G}$.

In Section 2 below, we argue that a fertile ground for building the games in $\mathcal{G}$ is to rely on the *implicit knowledge* of LLMs; this leads to the proposal to use Cross-Entropy Games (Xent Games) as a means from which to build $\mathcal{G}$.

1.4.2. *Meta-Game Design Elements.*

Our main contribution is the principled derivation in Section 4 of a meta-algorithm, called *cognitive training*, to build a useful curriculum of games $(G_k)_{k\geq 0}$ in $\mathcal{G}$ to develop the cognitive capabilities of a model $\mathcal{M}$, building a sequence of models where $\mathcal{M}_{k+1}$ is obtained by training $\mathcal{M}_k$ on G_k. The *cognitive training* algorithm is a greedy algorithm based on a *meta-objective* $\mathcal{O}$. Central to the definition of $\mathcal{O}$ is the concept of *transfer value* (Section 3.4.2), which estimates the relevance of learning a game to play another.

At each step $k \geq 1$, the choice of a new game H should balance:

- quality/relevance q: how much H improves performance on the old games $G_{<k} := G_1, \ldots, G_{k-1}$;
- diversity/novelty d: how much H brings skills that are not yet captured by the games $G_{<k}$;
- external benchmark performance b: how much H improves performance on external metrics;
- description length l: the raw code length of H.

In Section 4, we derive explicit principled definitions of each of the previous quantities, and derive a formulation for the meta-objective $\mathcal{O}$, namely

$$\mathcal{O} = \frac{qd + bp}{l},$$

where p is a *pressure* factor, balancing the relative importance of internal and external performance. Note that despite involving a priori a number of term that scales linearly in k (and similarly for memory), the meta-objective $\mathcal{O}$ satisfies some good scalability properties (see Section 4.3.5).

1.4.3. *Scope and Contributions.* The key contributions of this paper are the following:

- The focus on Problem 2, i.e. relevant skill discovery, with the idea that if general capabilities are achieved by curriculum learning, they automatically imply relevant skill discovery.
- A formal definition, building upon [HoEm25], of a space of tasks called *(streamlined) cross-entropy (xent) games*, endowed with a transfer value structure.
- A hypothesis that (streamlined) curricula made of xent games can approximate the learning of any curriculum; and hence that relevant skill discovery can be understood in terms of xent games and their transfer value structure.
- A formalization of greedy curriculum learning for xent games in terms of a meta-objective optimization.
- A derivation of an explicit formula for any meta-objective satisfying suitable consistency assumptions.

1.4.4. *Structure of the Paper.*

- In Section 2, we outline the key idea of implicit knowledge for an LLM: this suggests that much more can be done with an LLM than simple sampling.
- In Section 3, we present the space of games upon which cognitive training is based: the (streamlined) xent games, derived from the implicit knowledge of LLMs.
- In Section 4, we derive a natural framework for xent-game-based cognitive training for LLMs.
- In the same Section 4, we discuss several challenges and open questions associated with cognitive training.

2. Implicit Knowledge

If the models that we train are supposed to be useful for general tasks relevant to some world (e.g. human society), the games that they are trained on need to have a connection to that world. For language models, we posit that this connection is provided by the *implicit knowledge* of Large Language Models (LLMs) that have been trained on vast corpora of text data.

LLMs are generative models: given a context c, they define a conditional probability measure $\mathbb{P}_{\mathcal{M}}(\cdot \mid c)$ on sequences of tokens. For a given model $\mathcal{M}$, the likelihood is directly accessible: for any token sequence $x = x_1, \ldots, x_T$, the quantity $\log \mathbb{P}_{\mathcal{M}}(x \mid c) = \sum_{t=1}^{T} \log \mathbb{P}_{\mathcal{M}}(x_t \mid c, x_{<t})$ is readily available from $\mathcal{M}$. As a consequence, the information we can extract from $\mathcal{M}$ depends on how we use it: either as a generator, or as the basis for an algorithm relying on $\mathbb{P}_{\mathcal{M}}$ for e.g. perplexity comparison, search, and optimization in the space of token sequences.

In this section, we discuss the difference between:

- Explicit knowledge $\mathcal{E}_{\mathcal{M}}$: the information that can be quickly and reliably extracted from $\mathcal{M}$ by sampling answers from well-chosen prompts.
- Implicit knowledge $\mathcal{I}_{\mathcal{M}}$: the information that can be extracted by arbitrary algorithms with access to $\mathbb{P}_{\mathcal{M}}$.

Since generation is itself a (randomized) algorithm relying on $\mathbb{P}_{\mathcal{M}}$, we have $\mathcal{E}_{\mathcal{M}} \subset \mathcal{I}_{\mathcal{M}}$. In the following, we will see that there is a gap between the two kinds of knowledge: focusing on continuations allows one to explore only a tiny subset of observables (functions) of the measure $\mathbb{P}_{\mathcal{M}}$, while direct access to the model's measure allows for different procedures (search, contrastive experiments, ...) that further reveal the cognitive capabilities of the LLM. In Section 3, we exploit the implicit/explicit gap to define the Xent Games, designed to use implicit capabilities to improve the explicit knowledge of LLMs.

2.1. **Explicit Knowledge.** The *explicit knowledge* of an LLM is defined as follows: the set of questions and tasks for which, given a fixed computation budget, a model $\mathcal{M}$, typically fine-tuned for question answering, can provide a correct solution with sufficiently high probability by generating a continuation. This knowledge is essentially what a standard benchmark (e.g. one based on the samples generated by the LLM) would measure.

The notion of explicit knowledge has some important subtleties:

(1) The explicit knowledge for raw pre-trained models (not instruction-tuned) is ill-defined or poorly representative of the underlying knowledge of the LLM: for such models, the main objective is not to answer or to follow the output specifications. [1]

[1] With well-chosen prompts, one can steer LLMs towards the desired behavior, but this will not replace a good fine-tuning.

(2) Explicit knowledge depends on the decoding policy: it measures both the model distribution and an external choice of how we generate from it (greedy/sampling, temperature, ...).
(3) The answers of an LLM are stochastic and depend on sampling hyperparameters, so we allow for sampling a few answers before an algorithm produces the final output from these samples.

Finally, fine-tuning can modify part of the explicit knowledge: for instance, after alignment training, a model may refuse to answer a question its base version could have answered. This motivates the following perspective for understanding LLMs' capabilities: instead of asking the model what it knows, we can ask what can be extracted from the model probabilities.

2.2. **Implicit Knowledge.** We define the *implicit knowledge* as follows: the set of questions/tasks that can be answered algorithmically (given a fixed computation budget) given access to the model measure $\mathbb{P}_{\mathcal{M}}$. More specifically, we assume that we have access to the log-likelihoods of continuations $\log \mathbb{P}_{\mathcal{M}}(x \mid c)$. The notion of implicit knowledge is obviously appealing as it is the set of information that an LLM somehow already knows (in one form or another) from its pre-training.

The implicit knowledge differs from the explicit knowledge in at least three important ways. First, it is well defined for raw pre-trained models, as it does not rely on a question-answering setting. Second, it does not depend on the decoding policy as it depends solely on $\mathbb{P}_{\mathcal{M}}$. Third, it is, by its nature, related to a deterministic computational problem, rather than the consequence of a stochastic generator.

The simplest but most fundamental implicit knowledge is the ability to provide the log-likelihood of a sentence or continuation. For any $x = x_1 \ldots x_T$,

$$\log \mathbb{P}_{\mathcal{M}}\left(x_1 \ldots x_T \mid c\right) = \sum_{t=1}^{T} \log \mathbb{P}_{\mathcal{M}}\left(x_t \mid c, x_{<t}\right),$$

can be extracted from the model's measure, and is therefore part of the implicit knowledge. Yet, this quantity is not part of the explicit model: a model generally cannot explicitly report its correct value.

Of course, by sampling more and more continuations, we should theoretically be able to recover $\log \mathbb{P}_{\mathcal{M}}(x \mid c)$. Yet, the number of samples required is prohibitive. Indeed, for a fixed continuation x, observing x even once requires an order of $1/\mathbb{P}_{\mathcal{M}}(x|c)$ samples. Since $\mathbb{P}_{\mathcal{M}}(x \mid c)$ typically decreases exponentially with the length of x, it is impossible to recover the log-likelihood from samples within a reasonable computing time. This simple argument shows that having direct access to the log-likelihood is qualitatively different from having access only to the generation process.

2.3. **Examples of Implicit Knowledge of Pre-Trained Models.** Direct access to the log-likelihood enables procedures that are out of reach for explicit generation. As explained above, this is important for raw pre-trained models: even when a model does not reliably follow a question/answer format, its conditional measure can still be evaluated through log-likelihoods. In this section, we provide a non-exhaustive family of procedures for extracting implicit knowledge.

One way to do so is to consider the difference between log-likelihoods in which some candidate sequences x_1 or x_2 are inserted into possibly different contexts c_1 and c_2. This quantity measures how much more the model supports x_1 in the hypothetical world specified by c_1 than x_2 in the hypothetical world specified by c_2. By choosing contexts appropriately, and determining how to insert the candidate sequences into them, this likelihood difference can be used among other things to extract beliefs from a model, quantify the usefulness of some information, or define some contrastive objectives. In practice, contexts are built from a base context to which pre-prompts or sandwich prompts are added before and after it; we omit these details here, but they matter in concrete implementations.

2.3.1. *Likelihood difference for a fixed statement.* We can compare the likelihood of the same statement in two different contexts.

- **Truth-false difference.** For a statement s, we can consider

$$\begin{aligned} \Delta_{\mathrm{T/F}}(s \mid c) \quad &= \quad \log \mathbb{P}_{\mathcal{M}}(s \mid c, \text{"The following statement is true : "}) \\ &\quad - \log \mathbb{P}_{\mathcal{M}}(s \mid c, \text{"The following statement is false : "}) \end{aligned}$$

  A simple classifier predicts “true” whenever $\Delta_{\mathrm{T/F}}(s \mid c) > 0$. To reduce dependence on the exact prompt, we can compute the average of $\Delta_{\mathrm{T/F}}$ over paraphrases of the prefix and/or of the statement s. Closely related, in [KCAHK22], to study the calibration of models, they present a proposed answer to a model and ask it whether it is true or false, then read off the log-likelihood of the two options.

- **Multiple-choice selection for explanations.** More generally, given candidate explanations $c_1, \dots, c_n$, we can select $\arg\max_{i \in \{1,\dots,n\}} \log \mathbb{P}_{\mathcal{M}}(s \mid c_i)$. This allows one to select the most plausible explanation for s.

2.3.2. *Counterfactual thinking and surprise reduction.* A variant of the multiple-choice selection of explanations is to quantify the value of some information. Let q be a problem, x a reference solution, and h an additional piece of information. The difference

$$\Delta_{\text{info}}(h; q \to x) = \log \mathbb{P}_{\mathcal{M}}(x \mid q, h) - \log \mathbb{P}_{\mathcal{M}}(x \mid q)$$

measures how much h reduces the model's surprise about x. This makes it possible to quantify the usefulness of a piece of information without requiring the model to explicitly explain why (which actually does not provide a quantitative reliable number).

2.3.3. *Contrastive objective across models or checkpoints.* Another variant of log-likelihood comparison is to consider two different models or checkpoints. Given two models $\mathcal{M}_1$ and $\mathcal{M}_2$, the quantity

$$\Delta_{\text{ctr}}(x \mid c) = \log \mathbb{P}_{\mathcal{M}_1}(x \mid c) - \log \mathbb{P}_{\mathcal{M}_2}(x \mid c)$$

quantifies whether x is more plausible for $\mathcal{M}_1$ than for $\mathcal{M}_2$. When $\mathcal{M}_1$ is stronger, for example if it is a larger model, and $\mathcal{M}_2$ is weaker, this quantifies the continuations that are more interesting in the sense that a clever model can think about it, but a simpler model cannot. This is closely related to contrastive sampling methods (see e.g. [LHFLL22]).

2.3.4. *Verification vs construction.* Many tasks have solutions that are easier to verify than to construct.[2] A well-trained model can assign a higher likelihood to correct solutions than to incorrect ones, while still failing to construct a correct solution by direct generation.

Access to the log-likelihood transforms the verification task into a continuous signal: rather than a discrete "correct/incorrect" reward, $\log \mathbb{P}_{\mathcal{M}}$ can be used to construct a continuous objective that can guide search and optimization.

In addition, it is worth noting that some continuity in reward signals can be provided at the length level: rewards on incomplete responses can be elicited as well, and help steer the trained models towards high-reward responses.

This perspective can be useful to learn to solve problems that can even be a priori combinatorial in nature (by providing some suitable continuous relaxations of such problems using the LLM cross-entropy function to provide a "soft" enforcement of the constraints). The gap between verification and construction is one of the important gaps that can produce new data and games which can later be distilled into the explicit knowledge.

2.3.5. *Constrained optimization in the space of sequences and inverse prompting.* More generally, the log-likelihood (together with suitable contexts) can be used to define various losses on token sequences. A fundamental optimization problem is the prompting problem, which consists of finding the optimal set of tokens that satisfies some fixed constraints (format, length, vocabulary restrictions, ...) and maximizes the log-likelihood when inserted into a fixed context. This family includes some meaningful inverse problems:

Example 5 (Inverse prompting and summarization)**.** Given a fixed text x, we can search for a short sequence of tokens t such that x becomes highly likely when it follows t, i.e.

$$t^* \in \arg\max_{t \in \mathcal{C}} \log \mathbb{P}_{\mathcal{M}}(x \mid t),$$

where $\mathcal{C}$ encodes the constraints. With appropriate prompting between, before or after t and x, this can be interpreted as a form of summary: t summarizes x because it reduces the surprise of x for the model. This type of implicit knowledge is a direct building block of the subsequent Xent Games.

Example 6 (Common explanations)**.** A variant of the previous example can be obtained when given multiple texts $x_1, \dots, x_n$. We can search for a sequence of tokens t that makes them jointly more likely, but at the same time relatively unexpected from each of them individually

$$t^* \in \arg\max_{t \in \mathcal{C}} \log \mathbb{P}_{\mathcal{M}}(x_1, \dots, x_n \mid t) - \alpha \sum_{i=1}^{n} \log \mathbb{P}_{\mathcal{M}}(t \mid x_i).$$

With appropriate prompting, t can be understood as an interesting common feature about the texts.

[2]In the context of self-evaluation, the authors of [KCAHK22] note that "verification improves faster than generation quality in this context".

2.3.6. *Anomaly detection.* If a small part of a text has a high cross-entropy loss, it probably deserves attention: this could signal either an anomaly, an adversarial or injected segment, or simply a rare but meaningful information in the text that the model cannot well predict. This intuition has been used in practice for e.g. adversarial prompt detection [HWMHS23], outlier spatial trajectories [MPA24].

2.4. **Explicit vs Implicit Knowledge.** As discussed, the explicit knowledge of an LLM is a subset of its implicit knowledge, since sampling tokens from the model is an algorithm on its probability measure. It is a strict subset, because none of the probabilities in the examples in 2.3 can be computed with a (reasonable) amount of sampling. While some questions like the truth value of a statement can also be answered by generating tokens, having access to the probability allows us to determine the certainty of the model. This access through the model's probability measure furthermore does not require fine-tuning for instruction-following and is thus well-defined for pre-trained models.

Due to its vastness, generality and easy accessibility, we use the implicit knowledge of LLMs as the basis of Xent Games, introduced in next section. These games will make up the training curriculum of an agent designed for general capabilities.

3. Streamlined Xent Games

Building on the idea of implicit knowledge outlined in Section 2 above, we briefly outline the ideas behind Xent (Cross-Entropy) Games. Rather than fully specifying the language used to write Xent Games (see [HoEm25] for details), we focus on a core subset of them that we call *Streamlined Xent Games (SXGs)*, and we provide illustrative examples.

SXGs form a subspace of the Original Xent Games (OXGs) introduced in [HoEm25], which focuses on benchmarking. By contrast, SXGs are designed to make training more streamlined (i.e. more parallelizable, easier to optimize, and with extra differentiable structures [HERG26, HAGER26]) and therefore suitable for training models to play them. At the same time, it is reasonable to expect that the set of skills developed by SXGs is the same as that developed by the space of OXGs (see Section 3.3.2 below).

3.1. **Definition and Runtime.** Informally, Xent Games are text-based games played by some main players $\mathcal{M}$, with one or more LLMs used as "world models" (i.e. providing the environment dynamics and rewards). The whole space of games is endowed with some meta-data, corresponding to the specification of the models involved, which are each attached to a *model name* (i.e. they link variable names used in the game codes to concrete model checkpoints). In the simplest variant (on which we focus here), one of these is the main player model, which is the one under cognitive training; the other models are frozen and they play the roles of judge, data stream models, or NPCs (opponents / cooperators).

The game state consists of a collection of string registers that are updated according to basic string operations (Section 3.1.1), by using inputs from data streams, and by writing players' outputs.

The players receive as input strings read from the registers and produce output strings that are written back into the registers. In addition, they receive rewards based on signed cross-entropies of token strings stored in the registers (Section 3.1.2).

3.1.1. *Token String Space.* As text-based games, Xent games run on a space of strings, endowed with a small set of basic string operations. In the framework of OXGs, the allowed operations are: 'cat' (i.e. string concatenations at the character level) and 'cut' operations (i.e. splits between what comes before a first occurrence of a string and what comes after). In SXGs, the situation is simpler:

- String registers have a fixed token length, although different string registers may have different token lengths.
- Concatenations and cuts are replaced by copy operations made at the token level (rather than the character level), that copy part of a token string into another, stopping when the destination register is full.

3.1.2. *Xents and Xent Sums.* At the heart of xent games are cross-entropies of strings $\mathrm{xent}_{\mathcal{J}}$ computed by a (judge) model $\mathcal{J}$. If $x = x_1 \cdots x_n$ and $y = y_1, \ldots, y_m$ are token strings (read from a token string register), we define

$$\mathrm{xent}_{\mathcal{J}}(x|y) = -\log \mathbb{P}_{\mathcal{J}}(x|y) = -\sum_{i=1}^{n} \log \mathbb{P}_{\mathcal{J}}(x_i | y, x_{<i}).$$

Informally, $\mathrm{xent}_{\mathcal{J}}(x|y)$ measures how "surprised" the autoregressive model $\mathcal{J}$ is to see x after seeing y. We denote also $\mathrm{xent}_{\mathcal{J}}(x) = \mathrm{xent}_{\mathcal{J}}(x|\emptyset)$ where $\emptyset$ is the empty string.

The (judge) model $\mathcal{J}$ can be the main player model $\mathcal{M}$ itself (as in, e.g., pre-training games; see below), or a fixed pre-trained model, or a separate fixed agent. Depending on whether $\mathcal{J} = \mathcal{M}$ or $\mathcal{J} \neq \mathcal{M}$, the goal may be to improve the model used for the cross-entropy computation, or simply to optimize with respect to this cross-entropy defined by a fixed $\mathcal{J}$.

Remark 7. In practice, for SXGs, we clip the $\mathbb{P}_{\mathcal{J}}\left(x_i|y, x_{<i}\right)$ from below by $1/v$ where v is the vocabulary size (the number of possible tokens). This clips the per-token loss $-\log \mathbb{P}_{\mathcal{J}}\left(x_i|y, x_{<i}\right)$ from above by $\log v$. This prevents players to exploit very low probabilities of models to achieve high rewards: these log-probabilities are typically noisy and not meaningful (see the well-posedness criterion in [HoEm25]).

For a family of token strings $\{(x_j, y_j)\}_j$ and a family of models $(\mathcal{J}_j)_j$ a signed *xent sum* is an expression of the form

$$\sum_j \sigma_j \mathrm{xent}_{\mathcal{J}_j}\left(x_j|y_j\right), \qquad \sigma_j \in \{\pm 1\}$$

Xent sums can be used in two ways:

- as rewards given to the players;
- as skewed rewards (i.e. soft constraints, see next Section 3.1.3) given to the players.

3.1.3. *SXG: Definition and Basic Runtime Instruction.* An SXG consists of a sequence of moves of four types:

- *assign*: modify a string register using the token-string operations described in Section 3.1.1.
- *elicit*: request from a model a string of length n tokens.
- *reward*: reward a player based on a signed xent sum (evaluated on token-string registers).
- *ensure*: provide a smooth constraint on a player by imposing a soft positivity constraint $S \geq 0$ on a signed xent sum S (evaluated on token-string registers). Concretely, for a fixed $\lambda > 1$, the model receives reward S/λ if $S \geq 0$ or λS if $S < 0$ (see Remark 8 below).

Remark 8. The SXG *ensure* constraints follow the same "adversarial multiplier" logic as in OXG [HoEm25], but with a fixed finite multiplier $\lambda > 1$ in case of violation (and a small nonzero multiplier $1/\lambda$ in case of fulfillment). Natural examples of ensure constraints are true/false statements based on the implicit knowledge (as in Section 2.3.1). Another example of *ensure* constraint is based on the signed xent sum

$$\mathrm{xent}_{\mathcal{J}}(s2|s1 + \text{"comes after"}) - \mathrm{xent}_{\mathcal{J}}(s1|s2 + \text{"comes after"}),$$

which softly enforces that $s2$ is more likely to follow $s1$ than the reverse.

3.1.4. *Original Xent Game Language (OXGL).* The simple principles behind OXGs make them naturally suited to being expressed in a domain-specific language, which we call *OXGL* (Original Xent Game Language). The design is close in spirit to an assembly language: each line of code corresponds to one instruction of the game logic. Any program made of valid lines of code is valid (there are no conditional jumps), which allows for e.g. a batched execution for training. Each line corresponds to a basic instruction of the types listed in Section 3.1.3 above. In the OXGL specification [HoEm25], a number of instructions made to simplify the writing and reading of xent games by humans are added; however they do not alter the space of games being considered, and for simplicity, we will omit those here.

The inputs to the *reward* and *ensure* statements are signed xent sums. In OXGL, these instructions take three ingredients as inputs: the judge, the target string, and the (possibly empty) prefix strings.

3.1.5. *Streamlined Xent Game Language (SXGL).* As explained above, for the purpose of cognitive training, we rely on streamlined xent games (SXGs). This motivates the SXGL (Streamlined Xent Game Language) specification. SXGL is a minimalistic, assembly-like language, which only consists of two basic binary operators, $\ll$ and $\gg$, acting on three different types of objects:

- Models (main player, judges, data streams, NPCs). Each model has a prompt register and a score register, used to accumulate xent-based scores and to implement the ensure moves.
- Fixed-length token strings with insertion pointers, implementing the token-space operations described above (see Section 3.1.1 above).
- Cross-entropy register objects, which encapsulate a judge, a prefix and target strings. They are used to compute xent values throughout the game and to reward players by updating their score registers.

For example, given a player model $\mathcal{M}$, a judge model $\mathcal{J}$, a cross-entropy register $\mathcal{X}$, and a string register $\mathcal{S}$:

- $\mathcal{X} \ll \mathcal{J}$ sets $\mathcal{J}$ as the judge used by $\mathcal{X}$ in the following xent computations.

- $\mathcal{M} \ll \mathcal{X}$ computes the xent value defined by $\mathcal{X}$ (from its judge, prefix, and target strings) and adds it to the score register of $\mathcal{M}$, whereas $\mathcal{M} \gg \mathcal{X}$ subtracts it from $\mathcal{M}$'s score register.
- $\mathcal{S} \ll \mathcal{M}$ samples from $\mathcal{M}$ and writes the resulting token string into $\mathcal{S}$, whereas $\mathcal{M} \ll \mathcal{S}$ reads (part of) $\mathcal{S}$ and provides it as input to $\mathcal{M}$.

Since the two binary operators can act on an ordered pair of objects (of three possible types), this results effectively in $3 \times 3 \times 2 = 18$ types of operations, from which all operations needed to run xent games and to train models are needed. A detailed description of SXGL is given in Appendix A.

3.2. **Examples of Xent Games.** In this subsection, we review a number of xent games as a means of illustrating the potential of the space. While many xent games can be hand-designed for various purposes (see [HoEm25] for a few), our thesis is that the means towards achieving general capabilities is by relying on the automated design of such games.

3.2.1. *Pre-Training Game.* The classical pre-training objective, i.e. the minimization of $\mathrm{xent}_{\mathcal{M}}$ is a canonical Xent game. This game is in some sense the simplest: there is no move elicited from $\mathcal{M}$, just a reward $-\mathrm{xent}_{\mathcal{M}}(x)$ where x is a random string loaded from a data model.

Variants of the basic pre-training objective, as e.g. the multi-token prediction objective [Dee24] can naturally be represented as xent games.

3.2.2. *Reinforcement Learning as Pre-Training (RLP).* In [HAPCC25], a certain game, called RLP, is considered (inspired by a similar game [DDSW25], called RP). The RLP task is in fact a Xent Game. Changing the notation compared to the paper to fit the Xent game description, and removing a number of details that are not relevant to our discussion, the game is the following: given a game map $x_{<t}, x_t$ taken from a dataset (i.e. where x_t follows the tokens of $x_{<t}$), the main player $\mathcal{M}$ is elicited to produce a string c to improve the prediction of x_t given $x_{<t}$ and c: the reward of $\mathcal{M}$ is $-\mathrm{xent}_{\mathcal{M}}(x_t|x_{<t}, c)$. This is an interesting example of game where the judge model and the player are the same model.

3.2.3. *Distillation and Self-Distillation Games.* Distillation and Self-Distillation can both be viewed as instances of Xent Games.

In on-line distillation [LTML25, AVZSB24], we have a judge (teacher) model $\mathcal{J}$ that we want to distill into a player (student) model $\mathcal{M}$. The associated Xent game use the contrastive objective across models (Section 2.3.3). Concretely, we provide to $\mathcal{M}$ a context x (possibly void), and we elicit some answer c from $\mathcal{M}$. The reward for $\mathcal{M}$ is then $\mathrm{xent}_{\mathcal{M}}(c \mid x) - \mathrm{xent}_{\mathcal{J}}(c \mid x)$. Maximizing this reward corresponds to minimizing a reverse-KL objective between the (student) model $\mathcal{M}$ and the (teacher) model $\mathcal{J}$.

In the articles [HLBBG26, SDHA26], they do not consider the contrastive objective across models but the counterfactual thinking one (Section 2.3.2), where the extra information comes either from an exact proof, or feedback derived from previous attempts. A simplified Xent Game associated with [HLBBG26] can be described as follows. The main player $\mathcal{M}$ is cloned to obtain a judge $\mathcal{J}$ used to provide reward. We provide x to $\mathcal{M}$, and elicit a continuation c from $\mathcal{M}$. Next, conditioned on x, c we prompt $\mathcal{J}$ for feedback and obtain f. The player then obtains the reward $\mathrm{xent}_{\mathcal{M}}(c \mid x) - \mathrm{xent}_{\mathcal{J}}(c \mid x, f)$.

3.2.4. *Prompt Games.* Prompt games are our first concrete illustration of the idea of using implicit knowledge to improve the explicit behavior. Inspired by Example 5, the basic reverse prompting game is as follows. Consider a fixed judge $\mathcal{J}$; a game map consists of a text s. The player must find a prefix t such that, according to $\mathcal{J}$, s becomes likely after t: we reward the player with $-\mathrm{xent}_{\mathcal{J}}(s|t)$. This game is related to a number of tasks, including creative summarization or jailbreaking. Generalization follows from Example 6 by taking as game map a set of texts $s_1, \ldots, s_n$: the player must find t that makes them jointly more likely (for $\mathcal{J}$), while remaining relatively unexpected from each of them individually. Depending on the implementation details (constraints, prompting, regularization...), variants of this inverse prompting game have different interpretation, as discussed below.

- Creative summarization. If we add constraints that prevent token copying (length constraint, no common words, ...), a regularization term such as $-\mathrm{xent}_{\mathcal{J}}(t)/2$ (which can is equivalent to taking $-2\mathrm{xent}_{\mathcal{J}}(s|t) - \mathrm{xent}_{\mathcal{J}}(t)$) to obtain a well-formed text t, and insert some sandwich prompt between s and t, then the optimal prefix t can be considered as a summary of s.
- Prompt injection. Prompt injections can be formulated as Xent Games. A simple case is the static objective: find a prompt (or injected data) that causes a model $\mathcal{J}$ to generate a fixed output s, independent of the user's instructions i and data d. To evaluate the gap between the response generated by the LLM $\mathcal{J}$ and the target answer s, the authors of [LYZZX24] propose using a xent loss, e.g. $\mathrm{xent}_{\mathcal{J}}(s \mid i, d, x)$.

In the corresponding xent game, another model (the attacker) proposes x (conditioning on (i, d) or not), to maximize the expected reward over a training set of instructions-data pairs.

- Defensive variants. Defensive methods such as DataSentinel [LJJSG25] can be viewed as a two players Xent Game, whose goal is to prevent injected tasks. Two models, an attacker $\mathcal{A}$ and a detector $\mathcal{D}$, compete; this leads to a game-theoretic min-max optimization problem. The attacker outputs contaminated prompts or data (given some instruction and data to process) so that a judge LLM $\mathcal{J}$ executes an injected task instead of the intended task, while also avoiding to be detected. The detector receives a fixed detection instruction s_d and the (possibly contaminated) prompt/data and outputs a secret detection string which serves as marker. This marker should be present when the input is clean, and absent from the detector's output if contaminated. In this setup, all losses are cross-entropy terms. For example, detection is quantified by the cross entropy $\mathrm{xent}_{\mathcal{G}}(k \mid s_d, x)$ of the marker k given the detection instruction s_d concatenated with the potentially contaminated data x.

3.2.5. *Approximate Verifiable Reward Games.* When verification and scoring is easier than construction, a judge model can be used to enforce legality of player moves and score plausibility. As explained in Section 2.3.4, such well-trained judge, even if imperfect, can provide a continuous signal: reward in a verifiable task is thus turned into a continuous signal that can drive search optimization.

One may worry that the player will discover ways to exploit the judge biases rather than solve the intended verifiable task. While it is always possible to add an external reward for truly solving the problem, the Xent Games usage relies on transfer across games: if reward hacking is too easy, we conjecture that the model will not learn new generalizable capabilities, and such game will be filtered out when games are selected in the curriculum.

Examples of such verifiable reward games include e.g. chess and mathematical proofs (as discussed in [HoEm25]),

3.3. **Xent Game Space Properties.**

As argued in Section 3.2, the space of xent games $\mathcal{G}$ form a vast family, eliciting numerous skills highlighted in other papers. In this subsection, we argue that this space possesses a number of good properties that make it suitable for automated exploration towards building generally capable agents.

3.3.1. *Closure under Axioms.* The first natural property of the xent game space is its closure under natural axioms. In [HoEm25], the OXG space can be shown to naturally emerge from a small collection of game-theoretic and compositional axioms. The SXG space can be characterized very similarly:

Claim 9. If a space of text games on token string space (with the operations described in Section 3.1.1) contains the reverse prompt game (Section 3.2.4) and is stable under compositionality (we can append the instructions of one game to another), zero-summing (a player's loss can be a player's gain and vice versa) and adversarial rescalings (we can turn *reward* statements into *ensure* ones), then it must contain the space of SXG.

Remark 10. In [HoEm25], adversarial rescaling is formulated in terms of unbounded multipliers, leading to hard constraints; for the sake of training models (rather than evaluating them), it is better to work with soft constraints (with bounded multipliers); the principle is exactly the same. Similarly, the set of allowed operations on the string space in SXG is slightly different to that of OXG, but the principle is exactly the same, and if we use the operations described in Section 3.1.1, we obtain the xent games described in Section 3.1.3.

3.3.2. *Web-of-Games Assumption and Curriculum Universality.* A second important conjectural property of the xent game space outlined in [HoEm25] concerns the transfer value: informally, the idea is that it is relatively easy, given a set of games, to "discover" new, related games, allowing one to navigate in the space of games, and to grow the skills of a model. In the context of curriculum learning, this motivates the following approximation claim about general (i.e. not necessarily xent) game curricula:

Claim 11. Let E be a skill evaluation. Suppose there exists a curriculum of general games $\mathbb{G}_1, \ldots, \mathbb{G}_n$ such that training on this curriculum brings any model $\mathcal{M}$ in a family $\mathfrak{M}$ to a target skill level $\Lambda_{\mathcal{M}}$ on E. Then, for any $\epsilon > 0$, there exists

- a judge model $\mathcal{M}_{\mathcal{J}}$,
- a curriculum of xent games $G_1, \ldots, G_p$ of xent games (possibly defined using $\mathcal{M}_{\mathcal{J}}$) that is algorithmically computable from $\mathbb{G}_1, \ldots, \mathbb{G}_n$,

such that training any $\mathcal{M} \in \mathfrak{M}$ on $G_1, \ldots, G_p$ achieves skill level at least $\Lambda_{\mathcal{M}} - \epsilon$ on E.

Taken at face value, this claim can be seen as relatively straightforward: one can demand that $\mathcal{M}_{\mathcal{J}}$ be strong enough to evaluate the games of $\mathbb{G}_1, \dots, \mathbb{G}_n$ and approximate (naively) the scores by relevant xent combinations.

Similarly to the universality theorems for neural networks, the means to justify Claim 11 are not necessarily directly informative of practical/relevant uses of xent games; rather, the statement's point is that "nothing is missing" from the space.

In Section 4, we leverage these assumptions to formulate cognitive training as a meta-sampling process over the space of Xent Games.

3.4. **Game Training and Transfer.** The purpose of xent games is to provide a suitable environment for robust learning. In this subsection, we introduce the key notion of transfer value between games, which quantifies how training under a scheme Φ on one game affects the performance on another.

3.4.1. *Training Scheme.* We consider a fixed training scheme Φ, which defines, for any xent game G, a map $\mathcal{M} \mapsto \Phi_G \mathcal{M}$ which returns the model obtained by training $\mathcal{M}$ on G for *one run* of the game. A game with multiple training steps should be thought of in our framework as being made of many concatenated copies of a game $G \oplus G \oplus \cdots \oplus G$.

We assume that Φ is invariant by affine changes of coordinates, i.e. if $\alpha G + \beta$ denotes the games obtained by multiplying all scores by α and adding a constant β to them, then training is identical and we obtain $\Phi_{\alpha G + \beta} = \Phi_G$. This is the case for GRPO-style training based on centered and normalized rewards, as well as for algorithms such as [HERG26, HAGER26].

3.4.2. *Transfer Value.* A central quantity in our approach is the transfer value between games. Informally, the transfer value $\mathcal{T}_G^{\mathcal{M}}(H)$ encodes "how much in expectation does training on G teaches $\mathcal{M}$ about how to play H". We denote the expected score of $\mathcal{M}$ on H by

$$\mathrm{S}_{\mathcal{M}}(H) := \mathbb{E}\left[\mathrm{Score}_H(\mathcal{M})\right].$$

The transfer value is defined as follows.

Definition 12. For two games G and H, the transfer value $\mathcal{T}_G^{\mathcal{M}}(H)$ from G to H for $\mathcal{M}$ is

$$\mathcal{T}_G^{\mathcal{M}}(H) := \mathrm{S}_{\Phi_G[\mathcal{M}]}(H) - \mathrm{S}_{\mathcal{M}}(H).$$

More generally, if E is an external evaluation and G is a xent game, we define

$$\mathcal{T}_G^{\mathcal{M}}(E) := \mathrm{S}_{\Phi_G[\mathcal{M}]}(E) - \mathrm{S}_{\mathcal{M}}(E),$$

where $\mathrm{S}.(E)$ is the expected reward on E.

Remark 13. While these notions are well-posed theoretically, estimating them in practice may require a large number of sample or other techniques (such as the upcoming [HERG26, HAGER26]).

In Section 4 below, transfer value will be key to:

- estimating the internal relevance of candidate new games;
- estimating the novelty brought by new games compared to previously selected games.

3.4.3. *Positive Correlation.* A fundamental assumption upon which cognitive learning is that we can work with games that allow us to grow a curriculum constructively in a monotone way, i.e. without "needing to make a step back for every two steps forward".

For two Xent games $G_1, G_2 \in \mathcal{G}$, we say that they are nonnegatively correlated (relative to a model $\mathcal{M}$) if $\mathcal{T}_{G_i}^{\mathcal{M}}(G_j) \geq 0$ for $i, j \in \{1, 2\}$: informally speaking, this simply means that learning one doesn't make $\mathcal{M}$ worse at the other. To grow a curriculum of games in $\mathcal{G}$, we want to be able to pick at any time (and then optimize with respect to the meta-objective, see Section 4) new games that are nonnegatively correlated with respect to all the past games of the curriculum, as measured at appropriate times (see also Section 4.3 below for a justification of the times): if we have a built a curriculum $G_0, \dots, G_{k-1}$ yielding the models $\mathcal{M}_1, \dots, \mathcal{M}_k$, we want to be able pick a next game G_k such that (*a minima*) for each $j < k$, we have:

$$\mathcal{T}_{G_j}^{\mathcal{M}_j}(G_k) > 0, \tag{3.1}$$

$$\mathcal{T}_{G_k}^{\mathcal{M}_k}(G_j) > 0. \tag{3.2}$$

Definition 14. We define $\mathcal{G}_k^+ \subset \mathcal{G}$ to be set of Xent games that are positively correlated to $G_{<k}$.

An undesirable outcome in this case is that we end up in a *transfer cul-de-sac*, i.e that we are not able to further pick any new game (even a copy of an old one) satisfying (3.1 and 3.2).

Remark 15. Although this is a priori undesirable, artificially clipping transfer values from below by 0 can allow one to continue performing cognitive training without breaking the meta-objective $\mathcal{O}$.

3.4.4. *Relevant Skill Discovery.* An important desirable feature of a curriculum of games $(\mathcal{G}_k)_{k\geq 0}$, which motivates cognitive training is that of *relevant skill discovery*:

Definition 16. We say that a sequence of games $(G_k)_{k\geq 0}$ achieves *relevant skill discovery* on $\mathcal{M}$ if, when training a sequence of models $(\mathcal{M}_k)_k$, with $\mathcal{M}_0 = \mathcal{M}$ and $\mathcal{M}_{k+1}$ obtained from $\mathcal{M}_k$ by training on G_k,we achieve the following:

- The process keeps discovering new skills, i.e. finding new games that are not completely covered (in the sense of transfer value) by any $G_{<k}$ for a fixed k.
- It keeps growing on already discovered skills, i.e. for any G_j discovered, performance on G_j keeps improving as k increases.

4. Cognitive Training

In the previous section, we described the Xent Game Space $\mathcal{G}$, which we postulate provides a good coverage of the learnable skills of an LLM (see Section 3.3.2). Our goal is now to provide an algorithm to construct a curriculum of games in $\mathcal{G}$ for cognitive training, i.e. to build a "flywheel" that continuously (and autonomously) provides a stream of games that improve the model's general skills.

Remark 17. In this section, we focus on the training of *a single model* $\mathcal{M}$, although we may use other frozen (non-trained) models as judges or data streamers. Simultaneously training a family of models can be an interesting way to make cognitive training more efficient, but goes beyond the scope of this paper.

In this section, we describe a meta-objective to gauge the quality of a stream of games, based on elaborations of the evolution-based ideas of [HoEm25]. Given an initial model $\mathcal{M} = \mathcal{M}_0$, the goal is to evolve it into a sequence of models $\mathcal{M}_1, \mathcal{M}_2, \ldots$ that are more and more generally capable, where each update $\mathcal{M}_k \mapsto \mathcal{M}_{k+1}$ is obtained by training on a game G_k.

4.1. **Cognitive Training as a Greedy Algorithm.** The central question is *how to choose the next game in the cognitive training curriculum.* We propose to approach this as a greedy optimization problem, focusing on picking the *next game* to grow the curriculum. Thi

Remark 18. We focus on picking a greedy algorithm, as the task of optimizing a true value function on all curricula is likely completely untractable as the curriculum grows. We postulate that this greedy process is sufficient towards discovering new skills and growing towards general capabilities.

At each step, the goal is to find the best (most valuable) game that allows us to evolve the current model towards higher "cognitive" abilities, i.e. that has acquired "the most valuable new skills" between steps k and $k+1$.

The *meta-game at stage k* consists in a meta-sampler model $\mathcal{M}_{\mathcal{M}}$ tasked with outputting a Xent game G_k (in SXGL): the reward to $\mathcal{M}_{\mathcal{M}}$ is evaluated by a meta-objective $G_k \mapsto \mathcal{O}(G_k)$, which depends on $G_{<k}$ and $\mathcal{M}_{\leq k}$.

The question of determining a principled form for $\mathcal{O}$ is a priori a very underspecified question: assuming that the goal of the meta-game is to ascribe a value to games, this raises the question of whether there is a meta-meta-objective, and so on and so forth... it is not clear that such a problem can be resolved in a constructive fashion, i.e. without relying upon an infinite number of assumptions or parameters. [3]

4.2. **Meta-Objective Formula.** In Section 4.3, we show that (perhaps surprisingly) an explicit determination of $\mathcal{O}$ from purely qualitative consistency principles can be obtained, substantially constraining the space of "sound" meta-objectives to an explicit natural choice formula[4].

Given a history $G_{<k}$ and resulting models $\mathcal{M}_{\leq k}$, and restricting the domain to the positively-correlated games $\mathcal{G}_k^+$ (see Section 3.4.3), we find

$$\mathcal{O} = \frac{qd + bp}{l}, \tag{4.1}$$

where we have

[3]This situation echoes a bit a similar situation in field theory, where a Lagrangian may define an action of govern fields in space-time via e.g. a least-action principles, while there is no least-action principle that a Lagrangian would itself satisfy a priori.

[4]A somehow analogous situation appears in the so-called bootstrap program in conformal field theory.

- the *quality* term q is given by $q(H) = \left(\sum_{j<k} \mathcal{T}_H(G_j)\right)^{1-\delta}$ for a diversity hyper-parameter $\delta \in [0,1]$
- the *diversity* term d is given by $d(H) = \left(\mathcal{T}_H(H) / \sum_{j<k} \mathcal{T}_{G_j}(H)\right)^{\delta}$,
- the *benchmark* term b is given by $b(H) = \mathcal{T}_H(E)$ for an external benchmark metric term E,
- the *pressure* term $p \geq 0$ can be taken as a fixed hyperparameter or be adjusted in time (see Remark 19 below).
- the *length* term l is the raw code length of a game written in SXGL.

The derivation is presented in Section 4.3 below; in particular, the key result is the derivation of the Internal Meta-Objective Theorem derived in Section 4.3.2. Note that despite a linearly growing number of terms in k for q and d (suggesting quadratic scaling in k to run Cognitive Training), the meta-objective computation is only growing very slowly in k (see Section 4.3.5).

4.3. Meta-Objective Principles.

In this section, we derive an expression for the meta-objective $\mathcal{O}$ aimed assessing the value of a new game G_k, given a past curriculum $G_{<k}$ of "old games" used to train the model sequences $\mathcal{M}_{<k}$.

4.3.1. *General Principles.*

The following line of reasoning leads us to a principled expression for $\mathcal{O}$

- The goal of $\mathcal{O}$ is to measure the value that a new game brings towards training the next model.
- The $\mathcal{O}$ value combines *internal* and *external* relevance terms, denoted by $\mathcal{I}$ and $\mathcal{E}$ respectively.
- Both $\mathcal{I}$ and $\mathcal{E}$ terms measure the "useful work performed" if we train the model $\mathcal{M}_k$ on H.
- The $\mathcal{I}$ and $\mathcal{E}$ terms of a game H are to be normalized by its code length l in number of tokens.
- The $\mathcal{E}$ term depends on the difference on an external benchmark metric E if we train $\mathcal{M}_k$ on H.

We thus obtain the following expression for $\mathcal{O}$ of the form,

$$\mathcal{O} = \mathcal{I} + \mathcal{E},$$

in what follows, we will provide a principled derivation for $\mathcal{I}$ and $\mathcal{E}$.

4.3.2. *Internal Meta-Objective Derivation.*

To provide a principled derivation for the internal relevance $\mathcal{I}$ of the meta-objective $\mathcal{O}$ is quite nontrivial: the problem is a priori heavily underspecified. Assume again that we have obtained trained $\mathcal{M}$ on $G_{<k}$ leading to the sequence $\mathcal{M}_{\leq k}$.

(1) The term $\mathcal{I}$ depends on the past games $G_{<k}$ via the following transfer values:
  (a) The "new-to-old" transfer values $\mathcal{T}_H^{\mathcal{M}_k}(G_j)$ for $j < k$: this is the core of the *quality* measure.
  (b) The "old-to-new" transfer values $\mathcal{T}_{G_j}^{\mathcal{M}_j}(H)$ for $j < k$: this is the core of the *diversity* measure.
  (c) The "new-to-new" transfer value $\mathcal{T}_H^{\mathcal{M}_k}(H)$: this is a term that is useful for normalization.

(2) The term $\mathcal{I}$ should be factored as

$$\mathcal{I} = \frac{qd}{l},$$

where the *quality* q depends on $\left(\mathcal{T}_H^{\mathcal{M}_k}(G_j)\right)_{j<k}$ and the *diversity* d depends on $\left(\mathcal{T}_{G_j}^{\mathcal{M}_j}(H)\right)_{j<k}$ and on $\mathcal{T}_H^{\mathcal{M}_k}(H)$; using a product (rather than a sum) ensures the optimization of *both* quantities.

(3) The quality and diversity terms are invariant under *history fusion*: if for $0 < j < k$, we replace the steps

$$\mathcal{M}_{j-1} \xrightarrow[G_{j-1}]{} \mathcal{M}_j \xrightarrow[G_j]{} \mathcal{M}_{j+1}$$

by the training step (where [none] is an "idle" game, performing nothing)

$$\mathcal{M}_{j-1} \xrightarrow[G_{j-1} \oplus G_j]{} \mathcal{M}_{j+1} \xrightarrow[\text{[none]}]{} \mathcal{M}_{j+1},$$

then values of q and d for any G_k are left unchanged. [XXX : Fr : I would add somewhere a thing on $\mathcal{T}_{H_1 \oplus H_2}(G) = \ldots$ if you are ok]

(4) From this, we deduce that q must be only depend on the sum of transfer values, i.e. must be of the form

$$q(H) = f_q\left(\sum_{j<k} \mathcal{T}_H^{\mathcal{M}_k}(G_j)\right),$$

$$d(H) = f_d\left(\sum_{j<k} \mathcal{T}_{G_j}^{\mathcal{M}_j}(H), \mathcal{T}_H^{\mathcal{M}_k}(H)\right).$$

(5) Enforcing affine-rescaling invariance for $d(H)$ (see Section 3.4.2), we find that $d(H)$ can be written as a function

$$d(H) = f_d\left(\frac{\mathcal{T}_{G_k}^{\mathcal{M}_k}(H)}{\sum_{j<k}\mathcal{T}_{G_j}^{\mathcal{M}_j}(H)}\right)$$

for an increasing bijective function $f_d : \mathbb{R}_+ \to \mathbb{R}_+$.

(6) Using that $\mathcal{T}_{H\oplus H}(H\oplus H) \approx 4\mathcal{T}_H(H)$, and requiring $\mathcal{I}(H\oplus H) \approx \mathcal{I}(H)$, we obtain $qd(H\oplus H) \approx 2qd(H)$ (since $\mathcal{I} = qd/l$ and $l(H\oplus H) = 2l(H)$), and assuming that f_q and f_d must be homogeneous increasing functions, we find that their powers sum up to 1, yielding

$$\left(\sum_{j<k}\mathcal{T}_H^{\mathcal{M}_k}(G_j)\right)^{1-\delta}\left(\frac{\mathcal{T}_H^{\mathcal{M}_k}(H)}{\sum_{j<k}\mathcal{T}_{G_j}^{\mathcal{M}_j}(H)}\right)^{\delta},$$

for some *diversity parameter* $\delta \in [0,1]$.

From the above, we obtain the following:

Theorem (Internal Meta-Objective Theorem)**.** *A consistent internal meta-objective (according to the principles outlined above)* $\mathcal{I}$ *must be of the form*

$$\mathcal{I} = \frac{qd}{l},$$

where

$$q(H) = \left(\sum_{j<k}\mathcal{T}_H^{\mathcal{M}_k}(G_j)\right)^{1-\delta},$$

$$d(H) = \left(\frac{\mathcal{T}_H(H)}{\sum_{j<k}\mathcal{T}_{G_j}^{\mathcal{M}_j}(H)}\right)^{\delta},$$

where $\delta \in [0,1]$ *is a diversity hyper-parameter.*

Proof. [XXX : Fr : It is just above so maybe we even don't need this]Follows from the derivation in Section 4.3.2. ☐

4.3.3. *External Meta-Objective Derivation.* We can apply a similar idea to form a natural external meta-objective:

(1) Similarly to $\mathcal{I}$, the term $\mathcal{E}$ should satisfy $\mathcal{E}(H\oplus H) \approx \mathcal{E}(H)$.

(2) As a result, it is natural to postulate that

$$\mathcal{E} = \frac{bp}{l},$$

where $b = \mathcal{T}_H^{\mathcal{M}_k}(E)$, with E being the given external benchmark metric, and p denotes the *pressure* field, with $p(H\oplus H) \approx p(H)$.

(3) A principled determination for the pressure field p is again an under-determined problem in its own and beyond the scope of this paper.

Remark 19. A simple choice is $p = \eta T^{\chi}$ for hyper-parameters $\eta, \chi \geq 0$, where the (history fusion-invariant) "training time" T is defined as $\sum_{j<k} l(G_j)$. The choice $p = \eta T^{\chi}$ can be motivated by e.g. scaling law arguments, see e.g. [KMHBA20].

4.3.4. *Remarks on Derivation.* In Sections 4.3.1–4.3.3, a principled derivation for $\mathcal{O}$ is provided. A number of remarks are in order.

- The naming for the *quality* and *diversity* terms q and d obviously comes from the Quality-Diversity line of research in Artificial Life (see e.g. [PSS16, ECMO23]), but it is interesting to see that the derivation is made out of first principles only.
- The derivation can be viewed as the result of the answer to the question: what are the most natural objects we are able to use, and what is a consistent way to combine them? It is notable that a few natural constraints can narrow down the space of consistent meta-objectives to a few hyper-parameters.

4.3.5. *Scalability of Meta-Objective Computation.* While we assume an access to the whole set of archives $\mathcal{M}_j$ for $j \leq k$, it is remarkable that to compute the diversity denominator sum $\sum_{j<k} \mathcal{T}_{G_j}^{\mathcal{M}_j}(H)$ only requires us to measure the difference of performance of H on $\mathcal{M}_k$ and $\mathcal{M}$: summing the Definition 12, we get a telescopig sum, where the internal terms cancel.

Thus to perform Cognitive Training (Section 4.4), we only need to store the *latest archive* of $\mathcal{M}_k$. Similarly, to estimate the other sum $\sum_{j<k} \mathcal{T}_H^{\mathcal{M}_k}(G_j)$ only obvious requires access to $\mathcal{M}_k$, but also can be computed exactly in a time *linear in the number of different games* used (and furthermore is easily parallelizable).

4.4. **Cognitive Training Algorithm.**

In Section 4.3, we have derived the explicit form for the meta-objective $\mathcal{O}$ that, given an initial segment of the curriculum $G_{<k}$, ascribes the value to a new game $H \in \mathcal{G}_k^+$ (the space of Xent games positively correlated to $G_{<k}$, see Section 3.4.3). From this we first derive a cognitive training step:

Definition 20 (Cognitive Training Step)**.** Given a step k and a maximal length L, and a sequence of games $G_{<k}$ used to train a sequence of models $(\mathcal{M}_k)_{k\geq 0}$ with $\mathcal{M}_0 = \mathcal{M}$ and $\mathcal{M}_{k+1}$ being obtained by training $\mathcal{M}_k$, optimize the meta-objective value $\mathcal{O}(H)$ on games of length $l \leq L$.

Remark 21. Playing this game is a priori hard, as it involves an optimization on the space of all games. As a result, playing can only be imperfect, and can be hard to learn (see Section 4.5.1).

From there, the cognitive training is defined as follows:

Definition 22. Given a meta-sampler $\mathcal{M}_{\mathcal{M}}$ (possibly with some prompt), the Cognitive Training consists in the iteration of Cognitive Training Steps (see Definition 4.4) played by $\mathcal{M}_{\mathcal{M}}$.

Remark 23. Note that a priori, the cognitive training algorithm is not a meta-game itself (it has no single objective), being rather the result of iterated play; it is rather a greedy optimization algorithm.

4.5. **Outlook.**

4.5.1. *Playing the Meta-Game.* The question of playing the meta-game well, i.e. of how to train the meta-sampler $\mathcal{M}_{\mathcal{M}}$ to achieve strong performance on the meta-objective $\mathcal{O}$ is delicate and beyond the scope of this paper. While it is not clear a priori how strong a generalist pre-trained model needs to be to learn to play $\mathcal{O}$, there is (to the best of our knowledge) no game with verifiable rewards that are playable by a human that an LLM of current sizes (as of Spring 2026) cannot play. A natural setting to allow for the training is to increase the number of samples. This can be achieved by e.g. relying on faster transfer learning estimates to increase sample throughput (such as the upcoming [HERG26, HAGER26]). A promising way is to train small models $\mathcal{M}$ and use this as a baseline to transfer to larger models $\mathcal{M}$.

4.5.2. *Open-Ended Skill Discovery.* The most significant claim that can be made in the context of cognitive training that it can lead to an answer to Problem 2, i.e. to *relevant skill discovery* (in the sense of Definition 16 above). The claim is based on the following:

- On the absence of an examples of a verifiable-reward games that humans can play and that reasonably large language models cannot learn to play.
- On the implausibility that we can achieve very high value of qd having $\min(q,d)$ very small for a nontrivial range of δ-values $\Delta \subset [0,1]$.

Claim 24. There are reasonably large model $\mathcal{M}$ and meta-sampler model $\mathcal{M}_{\mathcal{M}}$ such that we have the following: for a nontrivial range $\Delta \subset [0,1]$ of δ-values, cognitive training with external pressure $p = 0$ yields unbounded relevant skill discovery.

Adding an external benchmark E for which a certain score level λ_E is achievable using a curriculum $\mathbb{G}$ (not necessarily consisting of xent games), an analogous question is raised.

Claim 25. Assume $(E, \lambda_E, \mathbb{G})$ is given. There are reasonably large model $\mathcal{M}$ and meta-sampler model $\mathcal{M}_{\mathcal{M}}$ (with access to $\mathbb{G}$ in its prompt), a nontrivial range $\Delta \subset [0,1]$ of δ-values, and a pressure schedule p, such that cognitive yields unbounded relevant skill discovery, while converging to λ_E performance on E.

Remark 26. The access to $\mathbb{G}$ for $\mathcal{M}_{\mathcal{M}}$ is not an ideal constraint (which can be voided in natural cases), but it is needed to avoid some situations where learning E involves access to e.g. a cryptographic secret.

4.6. **Numerical Simulation.**

- [Find appropriate transfer function] [Implement training curve: games that are too hard are no learnable] [Assume all lengths are the same]
- [Growing some game of skills]

5. Conclusion and Perspectives

In this paper, we have laid out the foundations of cognitive training, a principled process by which we iterate a greedy optimization step, called the *meta-game*, which consists of a *meta-objective.*

Our main result can be informally summarized as follows: if there is a simple and principled way to grow general capabilities via a greedy optimization method on a space of tasks naturally endowed with natural composition features (such as the space of Xent games), it has essentially to be cognitive training.

Cognitive training based on the meta-objective $\mathcal{O}$ seems hence to offer a compelling way to build models endowed with general capabilities.

Acknowledgements. Many insights presented in this paper have emerged from works in collaborations with Diego Dorn, Vassilis Papadopoulos, Marco Tuccio, Jérémie Wenger, and Nicolas Zlatoff on Large Language Models, with whom key ideas were discussed and investigated.

In addition, the authors would like to thank Emmanuel Abbé, Apoorv Agarwal, Mathieu Alain, Mohammad Asani, Alberto Bietti, Gloria Capano, Rahul Chalamala, Tarun Chitra, Jordan Cotler, Davide Crapis, Marco De Rossi, Mario Geiger, Evgenii Golikov, Alex Graves, Nicola Greco, Bara Hudcová, Arthur Jacot, Niels Linnemann, Tomáš Mikolov, Clément Moulin-Frier, Max Nye, Mihir Patel, João Penedones, David Pfau, Maciej Rudzinski, Stanislav Smirnov, Yi Sun, George Walker, Miles Wang, Jason Wang, Shouqiao Wang, and Matthieu Wyart for interesting discussions, as well as the participants to the Quine seminar and Demeco workshop for insightful questions and remarks.

Appendix: SXGL Specificatio

We provide a specification of SXGL (see Section 3.1.5). The language specification is deliberately minimal (even more so than that of OXGL defined in [HoEm25]), consisting of 18 instructions. A few preliminary remarks are in order:

- SXGL is suitable to reward (and thus train) several models; in the cognitive training described above, a single model is trained, and the rewards to other models should just be discarded by the interpreter.
- Some design elements can appear cumbersome to a human reader; the goal is not to make the code particularly human-readable, but to follow the design elements outlined in Section 3.1.5.
- There is no metadata associated with an SXGL program; SXGL programs can be seamlessly concatenated.
- Some syntactic sugar can be added to shorten the code without altering the instruction set; we avoid these questions here.

Global Metadata. An SXGL game space $\mathcal{G}$ consists of the following constant and global (i.e. common to all games $G \in \mathcal{G}$) metadata:

- a fixed token vocabulary $\mathcal{V}$;
- a list of models $\mathcal{M}^{(u)}$ for $u < U$, each based on $\mathcal{V}$ (with $\mathcal{M} = \mathcal{M}^{(0)}$ being the player model in cognitive training)
- a number $K \geq 1$ of token registers;
- a fixed common length parameter $L \geq 1$.

Code Structure.

- It is understood that any SXGL program starts with a newline symbol token (to enable seamless concatenation of programs); beyond this constraint, any token sequence is valid SXGL code.
- The interpreter will go line by line through the game code and interpret any line that matches the syntax of an *instruction line* (i.e. that satisfies the syntax of one of the 18 instructions below) as described.
- Otherwise, the code of the line is not interpreted (though it can be used to fill token string registers by further lines, as described below; instruction lines can also be used as data to fill token string registers).

Set of Variables. Given the global metadata, we define the following set of variables that are allowed to evolve during a game run:

- The token string registers s^a for $a < K$: for each $a < K$, we have $s^a \in \mathcal{V}^L$, i.e. s^a consists of L tokens in $\mathcal{V}$; each string s^a comes with a token caret register c^a splitting it into a "left half" ("before c^a") and a "right half" ("after c^a"); at initialization, c^a is such that the left half is empty.
- The xent object x (see below).
- Each model m_u for $u < U$ has a score register, which can be cleared.

Xent Object. The xent object x is used to as a "CPU" to compute the xents and to allow for string operations (there can be one or several xent objects, see below)

- The current context of each model $\mathcal{M} \in \mathfrak{M}$.

There is a judge model by default that is attached to x.

The 18 possible expressions.

- $x << x$: reward all models with the score registers and clear those (executed by default at the end of the program).
- $x >> x$: clears the prefix of the xent object
- $x << s$: make s the input of the xent object (typically whose xents are computed, but also useful for carets)
- $x >> s$: move the caret of s to the right by 1.
- $x << m$: make m the judge of the xent computation
- $x >> m$: apply ensure nonlinearity to m's score register, reward m with it, and clear the score register
- $s << x$: move the caret of s to the left by 1.
- $s >> x$: append what is on the right of s's caret to the prefix of the xent computation.
- $s << m$: elicit m into s to the left of the caret location.
- $s >> m$: reveal what is the left of the caret of s to m.
- $s_\ell >> s_r$:
  - if $s_\ell \neq s_r$: fill s_r's with tokens on the left of s_r's caret using tokens of s_ℓ that are on the left of s_ℓ's caret, and shift s_r's caret to the left. No overflow allowed.
  - if $s_\ell = s_r$ (i.e. $s >> s$): load the tokens before the instruction $s >> s$ into s to the left of the caret location, until that left of the caret is filled.
- $s_\ell << s_r$:
  - if $s_\ell \neq s_r$: fill s_ℓ with tokens from s_r from the caret and shift s_ℓ's caret to the right (and cut if there is overflow).
  - if $s_\ell = s_r$ (i.e. $s >> s$): load the previous chars into s from the left to the right of that line, and move the caret of s to the right.
- $m << x$: add the current xent value to the score register of m.
- $m >> x$: subtract the current xent value from the score register of m.
- $m << s$: reveal what is to the right of the caret of s to m.
- $m >> s$: elicit m into s from the left and until the caret.
- $m_\ell << m_r$:
  - if $m_\ell \neq m_r$: transfers the context of m_r to m_ℓ, and clear the context of m_r;
  - if $m_\ell = m_r$ (i.e. $m << m$): clears the context of m.
- $m_\ell >> m_r$:
  - if $m_\ell \neq m_r$: transfers the scores of m_ℓ to m_r and clears the score of m_ℓ;
  - if $m_\ell = m_r$ (i.e. $m >> m$) clears the scores of m.

References

[AVZSB24] R. Agarwal, N. Vieillard, Y. Zhou, P. Stanczyk, S. Ramos, M. Geist, and O. Bachem, On-policy distillation of language models: Learning from self-generated mistakes. In The Twelfth International Conference on Learning Representations, 2024.

[AAT2023] A.A. Team, J. Bauer, K. Baumli, S. Baveja, F. Behbahani, A. Bhoopchand, N. Bradley-Schmieg, M. Chang, N. Clay, A. Collister, V. Dasagi, L. Gonzalez, K. Gregor, E. Hughes, S. Kashem, M. Loks-Thompson, H. Openshaw, J. Parker-Holder, S. Pathak, N. Perez-Nieves, N. Rakicevic, T. Rocktäschel, Y. Schroecker, J. Sygnowski, K. Tuyls, S. York, A. Zacherl, and L. Zhang, Human-Timescale Adaptation in an Open-Ended Task Space, https://arxiv.org/abs/2301.07608

[Bax00] J. Baxter, A Model of Inductive Bias Learning, *Journal Of Artificial Intelligence Research*, **12**:149–198, 2000, https://arxiv.org/abs/1106.0245.

[BVLFW25] J. Beck, R. Vuorio, E.Z. Liu, Z. Xiong, L. Zintgraf, C. Finn, S. Whiteson, A Tutorial on Meta-Reinforcement Learning, A Tutorial on Meta-Reinforcement Learning, *Foundations and Trends in Machine Learning* **18**(2-3):224–384, https://arxiv.org/pdf/2301.08028.

[AFKKQH22] C. An, J. Feng, K. Lv, L. Kong, X. Qiu, X. Huang, CoNT: Contrastive Neural Text Generation, *Advances in Neural Information Processing Systems* **35** (NeurIPS 2022), https://arxiv.org/abs/2205.14690.

[BaSt07] B. Banerjee and P. Stone, General Game Learning using Knowledge Transfer, Proceedings of the 20th International Joint Conference on Artificial Intelligence, 2007. https://www.cs.utexas.edu/~ai-lab/pubs/IJCAI07-bikram.pdf

[BCK23] J. Blüm, J. Czech, and K. Kersting, AlphaZe∗∗: AlphaZero-like baselines for imperfect information games are surprisingly strong, Front. Artif. Intell., 12 May 2023, Sec. Machine Learning and Artificial Intelligence, Volume 6, https://doi.org/10.3389/frai.2023.1014561, 2023

[CGHCL21] S. Carré, F. Gabriel, C. Hongler, G. Lacerda, and G. Capano, Smart Proofs via Smart Contracts: Succinct and Informative Mathematical Derivations via Decentralized Markets, https://arxiv.org/abs/2102.03044.

[CGHCL24] S. Carré, F. Gabriel, C. Hongler, G. Lacerda, and G. Capano, Smart Proofs via Recursive Information Gathering: Decentralized Refereeing by Smart Contracts, *Distributed Ledger Technologies: Research and Practice*, **3**(1):1--19 https://dl.acm.org/doi/10.1145/3595298, 2024.

[CWWWX23] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, Xing Xie, A Survey on Evaluation of Large Language Models, https://arxiv.org/abs/2307.03109

[CZJGS24] W.-L. Chiang, L. Zheng, Y. Sheng, A.N. Angelopoulos, T. Li, D. Li, H. Zhang, B. Zhu, M. Jordan, J.E. Gonzalez, I. Stoica, Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference, https://arxiv.org/abs/2403.04132.

[Cho19] F. Chollet, On the Measure of Intelligence, https://arxiv.org/abs/1911.01547.

[CKKLP25] F. Chollet, M. Knoop, G. Kamradt, B. Landers, H. Pinkard, ARC-AGI-2: A New Challenge for Frontier AI Reasoning Systems, https://arxiv.org/abs/2505.11831

[Clu19] J. Clune, AI-GAs: AI-generating algorithms, an alternate paradigm for producing general artificial intelligence, https://arxiv.org/abs/1905.10985v2.

[CKATT18] M.-A. Côté, A. Kádár, X. Yuan, B. Kybartas, T. Barnes, E. Fine, J. Moore, R.Y. Tao, M. Hausknecht, L.E. Asri, M. Adada, W. Tay, and A. Trischler, TextWorld: A Learning Environment for Text-based Games, Proceeding of Computer Games Workshop, International Joint Conference on Artificial Intelligence 2018, https://arxiv.org/abs/1806.11532

[CoTh06] T.M. Cover, J.A. Thomas, Elements of Information Theory (2nd Edition), Wiley, 2006

[DAW24] B.C. Das, M. H. Amini, and Y. Wu, Security and Privacy Challenges of Large Language Models: A Survey, https://arxiv.org/abs/2402.00888.

[Dee24] DeepSeek-V3 Technical Report, DeepSeek-AI, https://arxiv.org/abs/2412.19437.

[DDSW25] Q. Dong, L. Dong, Y. Tang, T. Ye, Y. Sun, Z. Sui, F. Wei, Reinforcement Pre-Training, https://arxiv.org/pdf/2506.08007

[ECMO23] M. Etcheverry, B. W.-C. Chan, C. Moulin-Frier, P.-Y. Oudeyer, Meta-Diversity Search in Complex Systems, A Recipe for Artificial Open-Endedness? https://arxiv.org/abs/2312.00455

[GHWZ16] N. Ghani, J. Hedges, V. Winschel, P. Zahn, Compositional game theory, Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science. LICS '18. New York, NY, US: ACM. pp. 472–481. https://arxiv.org/abs/1603.04641

[GGKPW25] A. Gollakota, P. Gopalan, A. Karan, P. Peale, U. Wieder, When does a predictor know its own loss?, https://arxiv.org/abs/2502.20375.

[GBMMK17] A. Graves, M.G. Bellemare, J. Menick, R. Munos, K. Kavukcuoglu, Automated Curriculum Learning for Neural Networks, Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. https://arxiv.org/abs/1704.03003.

[HAPCC25] Ali Hatamizadeh, Syeda Nahida Akter, Shrimai Prabhumoye, Jan Kautz, M. Patwary1, M. Shoeybi, B. Catanzaro, Y. Choi, RLP: Reinforcement as a Pretraining Objective, https://arxiv.org/abs/2510.01265.

[HiVC93] G. E. Hinton and D. Van Camp, Keeping the neural networks simple by minimizing the description length of the weights. In Proceedings of the sixth annual conference on Computational learning theory, pp. 5–13. ACM, 1993. https://dl.acm.org/doi/10.1145/168304.168306

[HoEm25] C. Hongler and A. Emil, Cross-Entropy Games for Language Models: From Implicit Knowledge to General Capability Measures, https://arxiv.org/abs/2506.06832.

[HERG26] C. Hongler, A. Emil, A. Renard, F. Gabriel, Frost Scores and Cross-Entropy Games: Sample-Efficient Training, in preparation.

[HAGER26] C. Hongler, A. Renard, F. Gabriel, A. Emil, Higher-Order Frost Scoring Schemes, in preparation.

[HLBBG26] J. Hübotter, F. Lübeck, L. Behric, A. Baumann, M. Bagatella, D. Marta, I. Hakimi, I. Shenfeld, T. K. Buening, C. Guestrin, A. Krause. Reinforcement Learning via Self-Distillation, https://arxiv.org/pdf/2601.20802.

[HWMHS23] Z. Hu, G. Wu, S. Mitra, R. Zhang, T. Sun, H. Huang, V. Swaminathan, Token-Level Adversarial Prompt Detection Based on Perplexity Measures and Contextual Information, https://arxiv.org/abs/2311.11509.

[HDPHR24] E. Hughes, M.D. Dennis, J. Parker-Holder, F. Behbahani, A. Mavalankar, Y. Shi, T. Schaul, T. Rocktäschel, Open-Endedness is Essential for Artificial Superhuman Intelligence, *Proceedings of the 41st International Conference on Machine Learning*, PMLR **235**:20597-20616, 2024. https://arxiv.org/abs/2406.04268

[Hut05] M. Hutter, *Universal algorithmic intelligence: Sequential Decisions based on Algorithmic Probability.* Springer, Berlin, 2005.

[JTZA26] X. Ji, R. Tutunov, M. Zimmer, H. B. Ammar. Scalable Power Sampling: Unlocking Efficient, Training-Free Reasoning for LLMs via Distribution Sharpening. https://www.arxiv.org/pdf/2601.21590.

[KCAHK22] S. Kadavath, T. Conerly, A. Askell, T. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Hatfield-Dodds, N. DasSarma, E. Tran-Johnson, S. Johnston, S. El-Showk, A. Jones, N. Elhage, T. Hume, A. Chen, Y. Bai, S. Bowman, S. Fort, D. Ganguli, D. Hernandez, J. Jacobson, J. Kernion, S. Kravec, L. Lovitt, K. Ndousse, C. Olsson, S. Ringer, D. Amodei, T. Brown, J. Clark, N. Joseph, B. Mann, S. McCandlish, C. Olah, and J. Kaplan, Language Models (Mostly) Know What They Know, https://arxiv.org/pdf/2207.05221, 2022

[KMHBA20] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, D. Amodei, Scaling Laws for Neural Language Models, https://arxiv.org/abs/2001.08361.

[KGRMI22] T. Kojima, S.S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa, Large Language Models are Zero-Shot Reasoners, https://arxiv.org/abs/2205.11916

[Lal2015] K.N. Laland, T. Uller, M.W. Feldman, K. Sterelny, B.B. Müller, A. Moczek, E. Jablonka, and J. Odling-Smee, The extended evolutionary synthesis: its structure, assumptions and predictions, *Proc. R. Soc. B*, **282**: 20151019, 2015.

[LeHu07] S. Legg, M. Hutter, Universal Intelligence: A Definition of Machine Intelligence, *Minds & Machines*, 17(4):391--444, https://arxiv.org/abs/0712.3329, 2007.

[LeHu06] S. Legg, M. Hutter, A Formal Measure of Machine Intelligence, IDSIA Technical Report 10-06, arXiv:cs/0605024v1

[LeSt10] J. Lehman and K.O. Stanley, Efficiently evolving programs through the search for novelty, *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*, 2010, https://doi.org/10.1145/1830483.1830638.

[LeSt11a] J. Lehman and K.O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. Evolutionary Computation, 19(2):189–223, 2011.

[LeSt11b] J. Lehman and K.O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In Proceedings of the 13th annual conference on Genetic and evolutionary computation, pages 211–218. ACM, 2011.

[LBL23] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, B. Newman, B. Yuan, B. Yan, C. Zhang, C. Cosgrove, C. D. Manning, C. Ré, D. Acosta-Navas, D. A. Hudson, E. Zelikman, E. Durmus, F. Ladhak, F. Rong, H. Ren, H. Yao, J. Wang, K. Santhanam, L. Orr, L. Zheng, M. Yuksekgonul, M. Suzgun, N. Kim, N. Guha, N. Chatterji, O. Khattab, P. Henderson, Q. Huang, R. Chi, S. M. Xie, S. Santurkar, S. Ganguli, T. Hashimoto, T. Icard, T. Zhang, V. Chaudhary, W. Wang, X. Li, Y. Mai, Y. Zhang, Y. Koreeda, Holistic Evaluation of Language Models, https://arxiv.org/abs/2211.09110

[LHFLL22] X.L. Li, A. Holtzman, D. Fried, P. Liang, J. Eisner, T. Hashimoto, L. Zettlemoyer, M. Lewis, Contrastive Decoding: Open-ended Text Generation as Optimization, Association for Computer Linguistics, https://arxiv.org/abs/2210.15097

[LKKYW25] B. Liu, C. Jin, S. Kim, W. Yuan. W. Zhao, I. Kulikov, X. Li, S. Sukhbaatar, J. Lanchantin, J. Weston, SPICE: Self-Play In Corpus Environments Improves Reasoning. https://arxiv.org/pdf/2510.24684, 2025.

[LJJSG25] Y. Liu, Y. Jia, J. Jia, D. Song, N.Z. Gong, DataSentinel: A Game-Theoretic Detection of Prompt Injection Attacks, 2025 IEEE Symposium on Security and Privacy (SP), 2190-2208, 2025.

[LYZZX24] X. Liu, Z. Yu, Y Zhang, N. Zhang, C. Xiao, Automatic and Universal Prompt Injection Attacks against Large Language Models, https://arxiv.org/abs/2403.04957.

[LWLZD24] R. Liu, J. Wei, F. Liu, C. Si, Y. Zhang, J. Rao, S. Zheng, D. Peng, D. Yang, D. Zhou, A. M. Dai, Best Practices and Lessons Learned on Synthetic Data, COLM 2024, https://arxiv.org/abs/2404.07503

[LTML25] K. Lu and Thinking Machines Lab, "On-Policy Distillation", Thinking Machines Lab: Connectionism, Oct 2025.

[MPA24] J. K. Mbuya, D. Pfoser, A. Anastasopoulos, Trajectory Anomaly Detection with Language Models, *SIGSPATIAL '24: Proceedings of the 32nd ACM International Conference on Advances in Geographic Information Systems*, 208–219, https://arxiv.org/abs/2409.15366.

[MCV20] Clara Meister, Ryan Cotterell, and Tim Vieira. If beam search is the answer, what was the question? *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2173–2185. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.170. https://aclanthology.org/2020.emnlp-main.170

[MKSLH15] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518, 529–533 (2015). https://doi.org/10.1038/nature14236

[MJB15] T. Mikolov, A. Joulin, M. Baroni, A Roadmap towards Machine Intelligence, https://arxiv.org/abs/1511.08130v2.

[MSBLB17] M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, M. Bowling, DeepStack: Expert-Level Artificial Intelligence in No-Limit Poker, https://arxiv.org/abs/1701.01724.

[OAI2020] T.B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D.M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language Models are Few-Shot Learners, *Advances in Neural Information Processing Systems* **33**:1877–1901, https://arxiv.org/abs/2005.14165.

[PBRW99] L. Page, S. Brin, R. Motwani, and T. Winograd, The PageRank Citation Ranking: Bringing Order to the Web. *Technical Report. Stanford InfoLab.* http://ilpubs.stanford.edu:8090/422/

[PWH24] V. Papadopoulos, J. Wenger, C. Hongler, Arrows of Time for Large Language Models, International Conference on Machine Learning 2024. https://arxiv.org/abs/2401.17505, 2024.

[PBS16] E. Parisotto, J.L. Ba, R. Salakhutdinov, Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning, International Conference on Learning Representations, 2016, https://arxiv.org/abs/1511.06342.

[PSS16] J.K. Pugh, L.B. Soros, K. O. Stanley, Quality Diversity: A New Frontier for Evolutionary Computation, *Front. Robot. AI,* 3 - 2016 | https://doi.org/10.3389/frobt.2016.00040.

[PCDSC24] Ji-Lun Peng, Sijia Cheng, Egil Diau, Yung-Yu Shih, Po-Heng Chen, Yen-Ting Lin, Yun-Nung Chen, A Survey of Useful LLM Evaluation, https://arxiv.org/abs/2406.00936v1

[Schmi07] J. Schmidhuber. Gödel machines: Fully self-referential optimal universal self-improvers. *Artificial general intelligence*, pages 199–226. Springer, 2007. https://sferics.idsia.ch/pub/juergen/gmAGI.pdf

[Schmi10] J. Schmidhuber. Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990-2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230-247, 2010. IEEE, https://people.idsia.ch/~juergen/ieeecreative.pdf.

[SDHA26] I. Shenfeld, M. Damani, J. Hübotter, P. Agrawal, Self-Distillation Enables Continual Learning, https://arxiv.org/abs/2601.19897

[SYCYL24] Chufan Shi, Haoran Yang, Deng Cai, Zhisong Zhang, Yifan Wang, Yujiu Yang, Wai Lam, A Thorough Examination of Decoding Methods in the Era of LLMs, https://arxiv.org/abs/2402.06925

[SYSKZ24] Sanchit Sinha, Yuguang Yue, Victor Soto, Mayank Kulkarni, Jianhua Lu, Aidong Zhang, MAML-en-LLM: Model Agnostic Meta-Training of LLMs for Improved In-Context Learning, International Conference on Knowledge Discovery and Data Mining, 2024, https://arxiv.org/abs/2405.11446

[Sig23] O. Sigaud, G. Baldassarre, C. Colas, S. Doncieux, R. Duro, P.-Y. Oudeyer, N. Perrin-Gilbert, V.G. Santucci, A Definition of Open-Ended Learning Problems for Goal-Conditioned Agents, https://arxiv.org/abs/2311.00344.

[SHSH18] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, *Science*, 7 Dec 2018, **362**(6419):1140-1144, DOI: 10.1126/science.aar6404, 2018.

[SDHA26] I. Shenfeld, M. Damani, J. Hübotter, P. Agrawal. Self-Distillation Enables Continual Learning. https://arxiv.org/pdf/2601.19897.

[SWLL24] Y. Song, G. Wang, S. Li, B.Y. Lin, The Good, The Bad, and The Greedy: Evaluation of LLMs Should Not Ignore Non-Determinism, https://arxiv.org/abs/2407.10457.

[StBy19] Felix Stahlberg and Bill Byrne. On NMT search errors and model errors: Cat got your tongue? Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3356–3362. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1331. URL https://aclanthology.org/D19-1331.

[SSSJ25] Z. Stojanovski, O. Stanley, J. Sharratt, R. Jones, A. Adefioye, Jean Kaddour, Andreas Köpf, REASONING GYM: Reasoning Environments for Reinforcement Learning with Verifiable Rewards, https://arxiv.org/abs/2505.24760

[TKM23] Q. Tan, A. Kazemi R. Mihalcea, Text-Based Games as a Challenging Benchmark for Large Language Models, International Conference for Learning Representations, TinyPapers, 2023.

[Tur50] A. M. Turing, Computing Machinery and Intelligence. *Mind* **49**:433-460, 1950.

[WLCS19] R. Wang, J. Lehman, J. Clune, K. O. Stanley. POET: open-ended coevolution of environments and their optimized solutions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 142–151, ACM, 2019.

[WWZ24] Z. Wang, P. Wang, K. Liu, P. Wang, Y. Fu, C.-T. Lu, C.C. Aggarwal, J. Pei, Y. Zhou, A Comprehensive Survey on Data Augmentation, https://arxiv.org/abs/2405.09591

[WXSLD22] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou, Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, Proceedings of NeurIPS 2022, https://arxiv.org/abs/2201.11903.

[Wil92] R.J. Williams, Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning, *Machine Learning*, **8**(3-4):229 - 256, https://dl.acm.org/doi/10.1007/bf00992696.

[XDCGZ24] Z. Xi, Y. Ding, W. Chen, B. Hong, H. Guo, J. Wang, D. Yang, C. Liao, X. Guo, W. He, S. Gao, L. Chen, R. Zheng, Y. Zou, T. Gui, Q. Zhang, X. Qiu, X. Huang, Z. Wu, Y.-G. Jiang, AgentGym: Evolving Large Language Model-based Agents across Diverse Environments, https://arxiv.org/abs/2406.04151v1.

[XFLZZ25(2025)] X. Xing, Z. Fan, J. Lou, G. Li, J. Zhang , D. Zhang, PretrainZero: Reinforcement Active Pretraining, https://arxiv.org/pdf/2512.03442

[ZHLLC25] J. Zhang, S. Hu, C. Lu, R. Lange, J. Clune, Darwin Gödel Machine: Open-Ended Evolution of Self-Improving Agents, https://arxiv.org/abs/2505.22954.