Bias Amplification in Language Model Evolution: An Iterated Learning Perspective

Yi Ren
UBC
renyi.joshua@gmail.com

Shangmin Guo
University of Edinburgh
s.guo@ed.ac.uk

Linlu Qiu
MIT
linluqiu@mit.edu

Bailin Wang
MIT
bailin.wang28@gmail.com

Danica J. Sutherland
UBC & Amii
dsuth@cs.ubc.ca

Abstract

With the widespread adoption of Large Language Models (LLMs), the prevalence of iterative interactions among these models is anticipated to increase. Notably, recent advancements in multi-round on-policy self-improving methods allow LLMs to generate new examples for training subsequent models. At the same time, multi-agent LLM systems, involving automated interactions among agents, are also increasing in prominence. Thus, in both short and long terms, LLMs may actively engage in an evolutionary process. We draw parallels between the behavior of LLMs and the evolution of human culture, as the latter has been extensively studied by cognitive scientists for decades. Our approach involves leveraging Iterated Learning (IL), a Bayesian framework that elucidates how subtle biases are magnified during human cultural evolution, to explain some behaviors of LLMs. This paper outlines key characteristics of agents' behavior in the Bayesian-IL framework, including predictions that are supported by experimental verification with various LLMs. This theoretical framework could help to more effectively predict and guide the evolution of LLMs in desired directions. The code for experiments is available at https://github.com/Joshua-Ren/iICL.

1 Introduction

Recent large language models (LLMs) have shown remarkable instruction-following ability and an increasing number of applications; it is thus reasonable to expect they are likely to become more widespread. Moreover, *interactions* between LLMs (either multiple models, or different generations of the same model) may also become very commonplace in the near future. In fact, many recent works consider iterative on-policy self-data-augmentation solutions to break through the bottleneck of human-generated supervisions, e.g., self-instruct (Y. Wang et al. 2022), self-refine (Madaan et al. 2023), hypothesis refinement (Qiu et al. 2024), self-distill (C. Xu et al. 2023), self-instruct (Y. Wang et al. 2022), self-reward (Weizhe et al. 2024), self-feedback (W. Xu et al. 2024), RAFT (Dong et al. 2023), ReST (Gulcehre et al. 2023), iterated DPO (Xiong, Dong, Ye, Zhong, et al. 2023), OAIF (Guo et al. 2024), SPIN (Z. Chen et al. 2024), and many more. Whether the model's knowledge is updated through in-weight or in-context mechanisms, these methods involve an LLM learning from a corpus (comprising data examples or evaluations) generated by another LLM (or itself), and subsequently transferring this acquired knowledge to others. Looking towards the long term, the future Internet may (for better or worse) contain substantial portions of LLM-generated text, which will, in turn, be employed for training the subsequent generation of models. It thus seems important to begin studying what this process will mean for future models.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-1220

Although these self-improving methods demonstrate considerable improvements on various benchmarks, a systematic understanding of why they work and what is their limitations is still missing. Some analysis of knowledge distillation might bring insights (Mobahi et al. 2020), as learning from data generated by another model is a type of distillation. But precisely analyzing the LLM's behavior on specific samples becomes increasingly difficult as it grows more complex. Instead, a behavioral-level analysis might be fruitful, akin to how the Bayesian framework can aid in comprehending the human cognitive system (T. L. Griffiths et al. 2023). By conceptualizing the LLM as an intelligent agent, we can draw parallels between its behaviors and the cultural evolution observed in humans. Iterated learning (IL), a framework proposed to study the evolution of knowledge and beliefs through a chain of learning among Bayesian agents (Kirby et al. 2007), stands out as a promising candidate for achieving our goals.

In this paper, we start by introducing the Bayesian-IL framework, demonstrating that agents engaged in such a process gradually amplify bias in their priors. This amplification process can be steered by introducing an interaction phase that "filters" or "re-ranks" the messages generated by the agents. Next, we theoretically justify that the in-context behavior of LLMs can be approximated by a Bayesian update, establishing a crucial link to the LLM system. To validate our claims, we conduct numerous experiments across different settings. Depending on the beneficial or detrimental nature of the bias, we propose various strategies to guide the evolution of LLM. The key contributions of this work are: 1) establishing the first Bayesian analysis of the full interactive learning process (including an interaction phase); 2) applying this framework to LLM agents and describing their evolution theoretically; 3) validating the theory and demonstrating how to guide LLM's evolution using experiments. We believe that our analysis can enhance our understanding of LLMs, and aid in designing more effective algorithms for alignment, bias mitigation or amplification, and similar tasks.

2 Background and Related Work

2.1 Iterated Learning

Iterated learning (IL) is a hypothetical procedure to simulate *how the tendency of specific properties of human culture or language gradually emerges and becomes dominant.* It is based on studying the behaviors of a chain of intelligent agents. From the perspective of an individual agent, the process involves initially acquiring knowledge from its predecessor (*imitation*), refining its beliefs while using them to conduct tasks (*interaction* with the world), and subsequently imparting its knowledge to the agents in the next generation (*transmission*). Cognitive scientists have applied this framework to explain various evolutionary phenomena of human society, including the emergence of compositionality in human language (Kirby et al. 2015), patterns in human object categorisation (T. L. Griffiths et al. 2008), and the evolution of color naming systems (Carlsson et al. 2023). The framework has also seen recent success with neural network agents, including in emergent communication (Guo et al. 2019; Ren et al. 2020), machine translation (Y. Lu et al. 2020), visual question answering (Vani et al. 2021), large vision-language models (Chenhao et al. 2024), and representation learning (Ren et al. 2023), indicative that this framework could also be useful for more general deep learning systems, like LLMs.

2.2 Related On-policy Self-data-augmentation Methods

While the theoretical guarantees for the Bayesian-IL framework studied in this paper rely on several assumptions, we posit that the behaviors observed for many recent "iterative self-data-augmentation" methods in LLM can be at least partially explained by the theory. We will now overview the basic assumptions, and how they fit with recent LLM approaches (more discussions in Appendix A).

First, the theory assumes "self-evolution," where all agents in different generations share the same initial knowledge. Methods like self-refine (Madaan et al. 2023) and hypothesis refinement (Qiu et al. 2024), which require the LLM to refine its output by the feedback from an identical LLM for several rounds, satisfy this assumption. Self-distill (C. Xu et al. 2023) and self-instruct (Y. Wang et al. 2022), if the models involved in different generations are the same, do as well. On the contrary, the super-alignment setting (Burns et al. 2023), where a stronger model is trained using the data generated by another weaker model, do not strictly fit with our analysis. However, if all the models are trained using a similar corpus, so that their initial knowledge should be similar, our analysis might still hold partially.

The theory also assumes the information transferred among agents is in the form of data examples, as in RAFT (Dong et al. 2023) and ReST (Gulcehre et al. 2023). Both methods consider a multi-generation data-transferring process, during which the bias is introduced by re-ranking the transferred data. Methods like self-reward (Weizhe et al. 2024) and self-refine (Madaan et al. 2023), which requires one agent to evaluate another agent's response, do not directly fit this assumption. However, if we also consider the evaluation as part of the data generated by the agent, the Bayesian-IL framework can still bring some insights. Furthermore, as analyzed in Ren and Sutherland (2024) that many preference alignment methods like direct preference optimization (DPO, Rafailov et al. (2024)) will naturally amplify the preference hidden in the pretrained model's prior. Then, those multiple-generation DPO variants, e.g., iterative DPO (Xiong, Dong, Ye, Z. Wang, et al. 2023), might face a more serious risk of amplifying malicious bias.

In summary, although the assumptions of our Bayesian-IL framework might not be satisfied by all practical algorithms, the general trends depicted by it, e.g., the bias amplification, the necessity of a good interaction phase, etc., would still hold. Please refer to Appendix A for more discussions.

3 Bayesian Analysis of Iterated Learning

3.1 Notations and Basic Behaviors of Bayesian Agents

We denote a data pair as $d = (x, y)$, where $d \in \mathcal{D} = \mathcal{X} \times \mathcal{Y}$, with $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. The (x, y) pair can be question and answer in a QA problem, the input and label in a supervised setting, or any type of prompt and output for in-context learning. The hypothesis $h \in \mathcal{H} : \mathcal{X} \to \mathcal{Y}$ describes the mapping between all possible x and their corresponding y. Note that h can be either explicit or implicit, depending on the task. For instance, in inductive reasoning, h represents the rule determining the output from input examples and is explicit, as the model can directly generate it using natural language. Conversely, in self-data-augmentation, where x is a topic and y is a paragraph generated based on x, h is likely to be implicit. In this context, h can be highly abstract with varying interpretations, such as the level of conciseness, helpfulness, or even the writer's preference for using rhyme.

Consider a general Bayesian agent whose behavior can be depicted by two basic procedures: *learning* and *sampling*. Learning involves updating the agent's knowledge based on observations, while sampling is a procedure wherein the agent generates data based on its knowledge. In this context, the agent's knowledge is encapsulated by its posterior over the hypotheses, i.e., $P_{lm}(h)$.

In Bayesian learning, we assume the agent holds a prior $P_0(h)$ at the beginning. Its posterior after observing $\mathbf{d} = (x_i, y_i)_{i=1}^N$ is calculated as

$$P_{lm}(h) = P(h \mid \mathbf{d}) \propto p(\mathbf{d} \mid h) \cdot P_0(h), \tag{1}$$

where $p(\mathbf{d} \mid h)$ is the likelihood of these N data pairs under a specific h; this is usually hard to calculate in practice.

Assume the agent holds a posterior $P_{lm}(h)$ during sampling. Then, given the input signal x, we can sample the corresponding $y \sim P_{lm}(y \mid x)$. Based on the fact that h determines the relationship between x and y, the above sampling procedure is equivalent to $y \sim \mathbb{E}_{h \sim P_{lm}(h)}[p(y \mid h, x)]$, which can be rewritten as $h \sim P_{lm}(h)$; $y \mid h \sim p(y \mid h, x)$. Following the definition of d and the assumption that x is uniformly distributed, the sampling procedure above is equivalent to $d \sim P_{lm}(d) \propto p(d \mid h) \cdot P_{lm}(h)$. If we instead first decide the most probable h rather than sampling h from the agent (maximum a posteriori (MAP), as is perhaps common subconscious behavior for humans), we then generate d by

$$d \sim p(d \mid h^*), \quad h^* = \operatorname*{argmax}_{h \in \mathcal{H}} P_{lm}(h). \tag{2}$$

3.2 Iterated Learning of Bayesian Agents

Iterated learning is a hypothetical process simulating how human language gradually evolves to become more efficient when transferred and utilized across generations. Typically, iterated learning repeats of the following three phases, as illustrated in Figure 1: an *imitation phase*, where an ignorant agent learns from the data generated by its predecessor; an *interaction phase*, where this agent uses the knowledge to accomplish the task, and hence refine its knowledge; and a *transmission phase*,

https://doi.org/10.52202/079017-1220

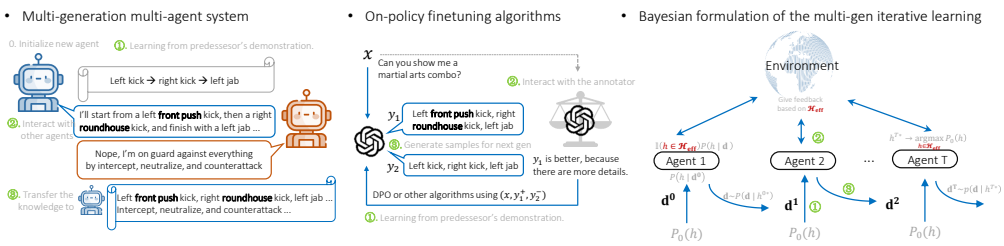

Figure 1: Examples of practical LLM systems that require knowledge transfer among different generations and how we use Bayesian agents to approximate their behaviors. ①, ②, and ③ denotes the imitation, interaction and transmission phases respectively.

where this agent generates useful data for the next generation. Combing with Section 3.1, we can get a picture of how h and d evolve as follows.

Initialization: at the beginning of the tth generation, a new agent$_t$, whose belief on h follows a prior distribution $P_0(h)$, is initialized. In lab experiments, $P_0(h)$ represents the belief of a well-educated participant who has not been previously involved in the target experiment. In in-context learning, a well-trained LLM also holds a complex and informative $P_0(h)$ based on the enormous corpus it is trained on and the task instructions in the prompt.

Imitation phase: after initialization, agent$_t$ then updates its knowledge by observing N data samples $\mathbf{d}^{t-1}$. Following the above learning procedure, the model's posterior should be $P(h \mid \mathbf{d}^{t-1})$.

Interaction phase: in this phase, the agent will accomplish specific tasks to refine its knowledge. The tasks involved in this phase can be diverse and complex. For example, in lab experiments (Kirby et al. 2015) and emergent communication (Ren et al. 2020), the agent plays a Leiws referential game (Lewis 2008) to rule out hypotheses representing a non-bijection between $\mathcal{X}$ and $\mathcal{Y}$; in representation learning (Ren et al. 2023), the agent directly conducts supervised learning on the downstream task to inhibit insufficient representations. Although it is hard to precisely formalize the behavior of the agent under these tasks precisely, their goals are consistent: we expect to "rule out" unsuitable hypotheses with carefully designed interactions. In an idealized setting, we might expect the agent's posterior to become proportional to $\mathbb{1}(h \in \mathcal{H}_{\text{eff}})P(h \mid \mathbf{d}^{t-1})$, where $\mathbb{1}(\cdot)$ is an indicator function and $\mathcal{H}_{\text{eff}} \subset \mathcal{H}$ is the subset of hypotheses that can accomplish the tasks. Broadly speaking, refining h or filtering $\mathbf{d}^t$ using the feedback from humans, LLM, or the environment, which is common in the aforementioned iterative self-data-augmentation methods, is also a type of task implicitly constraining $h \in \mathcal{H}_{\text{eff}}$.

Transmission phase: agent$_t$ now comes to the transmission phase, where it generates multiple data samples $\mathbf{d}^t$ for the next generation. The agent will first select the most probable hypothesis based on its current belief and then generate data samples, i.e., $d_i^t \sim p(d \mid h^{t*})$, where $h^{t*} = \text{argmax}_{h \in \mathcal{H}_{\text{eff}}} P(h \mid \mathbf{d}^{t-1})$. This accomplishes one generation of iterated learning.

3.3 Amplifying Biases is a Double-Edged Sword

In iterated learning, we repeat the phases mentioned above to get better h and $\mathbf{d}$. The limiting behavior of this process can be described by the following proposition. In short, the bias in $P_0(h)$ is guaranteed to be amplified generation-by-generation. Imposing appropriate $\mathcal{H}_{\text{eff}}$ (mainly through a carefully designed interaction phase) can control it.

Proposition 1. *Consider several Bayesian agents sharing the same prior $P_0(h)$ are conducting iterated learning for T generations. If T is sufficiently large, any agent$_t$ with $t > T$ will have*

$$P_{lm}(h) \to \mathbb{1}(h = h^{T*})$$

where h^{T*} is a stationary point (e.g. a local maximum) of $P_0(h)$ subject to $h \in \mathcal{H}_{\text{eff}}$.

To prove this, we first analyze iterated learning without the interaction phase. By drawing parallels between IL and EM (expectation-maximization) algorithms, we can prove that h^{T*} converges to the h with the maximum prior probability. We then consider the interaction phase, which introduces a "selection" pressure to rule out those $h \notin \mathcal{H}_{\text{eff}}$. By proving this process does not break the converging behavior of a non-interacting iterated learning, we achieve this proposition. (Please refer to Appendix B for more details.)

This proposition describes an inevitable bias amplification procedure as long as the model keeps learning from the data sampled from itself (via Bayesian update, distilling, or imitating, as long as the learning increases the model's confidence in these samples). However, in practical applications, we should bear in mind that bias amplification is a double-edged sword. If this bias is beneficial, like the simplicity bias in compositional language experiments, IL will help the model generate the "correct messages" more robustly. Imagine if we only have two possible hypotheses, i.e., h_{good} and h_{bad}, where $P_0(h_{good})$ is slightly larger than $P_0(h_{bad})$. Then by sampling $y \sim p(y \mid h, x) \cdot P_0(h)$, half of the chance we will get incorrect y coming from h_{bad}. Although we can get $y = \text{argmax}_y\, p(y \mid h, x) \cdot P_0(h)$ by using an extremely small temperature, the diversity of y provided by the likelihood will then disappear, which is not what we expected. How could we get samples that are both diverse and correct? Iterated learning can help with this problem by providing a posterior where $P_{lm}(h_{good}) \gg P_{lm}(h_{bad})$. With this posterior, sampling from $p(y \mid h, x) \cdot P_{lm}(h)$ would be similar to sampling from $p(y \mid h_{good}, x)$, which solves our problem.

Conversely, amplifying bias also negatively influences the system in several ways. Besides the cases where the bias is malicious (it can be solved by designing an appropriate interaction phase where $h_{bad} \notin \mathcal{H}_{\text{eff}}$), it also influences the model's creativity. Imagine we have multiple good h where $P_0(h_{g1}) > P_0(h_{g2}) > P_0(h_{bad})$, then IL will let us lose h_{g2} even its prior is only slightly smaller than h_{g1}. Such a mode decay phenomenon is quite similar to the "recursion curse" mentioned in (Shumailov et al. 2023): a more peaky $P_{lm}(h)$ will make those non-dominating h have a smaller probability, hence it is harder to keep these modalities during evolution. Touvron et al. (2023) also mentioned that iteratively fine-tuning would harm the creativity of the model. The solution could be early stopping the iterated learning or manually introducing more y that comes from h_{g2} during imitation.

In summary, to guide the LLM to self-evolve in an expected direction, we need a good $P_0(h)$, a carefully designed interaction phase, and an appropriate evolving time.

4 LLM-based Agents in Iterated Learning

4.1 LLM Behaves like a Bayesian Agent when Sampling

To transfer the Bayesian-IL analysis to LLM, we start by showing that the sampling and learning behaviors of an LLM agent can be depicted by Bayesian inference, following a few-shot in-context learning (ICL) scenario demonstrated in (Xie et al. 2022). In this setting, the message feed to the agent would be an instruction prompt $\mathbf{w}$ followed by N examples, i.e., $\mathbf{d}_N = (x_i, y_i)_{i=1}^N$. In other words, sampling y given the prompt, the examples, and the question x_{test} can be represented as:

$$y \sim P_{lm}(y \mid x_{\text{test}}, \mathbf{d}_N, \mathbf{w}) \triangleq P_{lmw}(y \mid x_{\text{test}}, \mathbf{d}_N), \tag{3}$$

where P_{lmw} is the model's belief conditioned on the instruction $\mathbf{w}$. If we call $\mathbf{d}_N$ as $\mathbf{d}^{t-1}$ (i.e., assume the examples are generated by agents in the previous generation) and assume the test question x_{test} is uniformly distributed, sampling new data based on instruction and few-shot examples can be expressed as $d^t \sim P_{lmw}(d \mid \mathbf{d}^{t-1})$, which is similar to the transmission phase in iterated learning.

Formally linking ICL and Bayesian-IL poses a non-trivial challenge, however, because the theoretical guarantee of Bayesian-IL relies on obtaining the MAP estimate of h at each generation. This is not immediately evident in ICL. Inspired by Xie et al. (2022), we first de-marginalize this posterior predictive distribution using the latent variable h, and then achieve the following proposition:

Proposition 2. *Consider that agent A is conducting in-context learning. If the prompt examples in $\mathbf{d}^{t-1}$ are generated by another agent B with the same prior knowledge (e.g., they come from the same checkpoint and use the same prompt), sampling from the posterior predictive distribution of agent A, i.e., $d^t \sim P_{lmw}(d \mid \mathbf{d}^{t-1})$, can be decomposed into: 1.) $h^{t*} \to \text{argmax}_h\, P_{lmw}(h \mid \mathbf{d}^{t-1})$, and 2.) $d^t \sim P_{lmw}(d \mid h^{t*})$, where h is a hidden variable that describes the mapping between x and y.*

The proof is in Appendix B.3. This proposition bridges LLM and Bayesian agents using its in-context behavior, which we believe is a ubiquitous procedure in any LLM system, irrespective of the subsequent information-updating strategy or the final target. For example, in an LLM-agent system, where no in-weights update exists, the model interacts with other agents (e.g., the human, the internal block of an LLM agent, or the environment) by generating responses based on the prompt and dialog

https://doi.org/10.52202/079017-1220

history. For LLM's finetuning, where various parameter updating strategies exist, the model also generates responses given the prompts, which is well depicted by the in-context behavior. Although the assumptions in this proposition do not exactly hold for all LLM systems, we believe our analysis can still roughly depict important trends of them. Please refer to Appendix A for more discussions.

4.2 LLMs in Different Algorithms have a Similar Target to Bayesian Learning

We then check the learning procedure. First, in a pure in-context learning setting like self-instruct (Y. Wang et al. 2022), self-refinement (Madaan et al. 2023), hypothesis search (Qiu et al. 2024), etc., the learning can be modeled by calculating the posterior $P_{lmw}(h \mid \mathbf{d}^{t-1})$, which is identical to the Bayesian learning discussed previously. Then, for those algorithms that require in-weights updates, like self-reward (Weizhe et al. 2024), self-play instruction tuning (Z. Chen et al. 2024), iterative DPO (Xiong, Dong, Ye, Z. Wang, et al. 2023), etc., the LLM might update its $P_{lmw}(h)$ using different loss functions. However, as all of these methods contain a procedure that encourages the models to increase their likelihood of the training samples generated by their predecessors, we should expect $P_{lmw}(\mathbf{d}^{t-1})$ to be increased after learning. As a result, the equivalent posterior $P_{lmw}(h)$ will implicitly favor those h that can generate $\mathbf{d}^{t-1}$, which aligns with the Bayesian learning target.

5 Experimental Verifications when the Hypothesis is Explicit

We directly verify our analysis above using an inductive reasoning task called Abstract Causal REasoning (Chi Zhang et al. 2021, ACRE), where all LLM agents update their knowledge via ICL. In this task, the model needs to infer and generate the shared rule by summarizing several input-output pairs. Specifically, assume there are M different objects, say `[A,B,C]`. One data pair $d = (x, y)$ is composed of an input x, i.e., a list of a subset of these objects, and an output y that represents the status of the light (could be on, off, or undetermined). In this experiment, the existence of a specific object triggers the light to be on. The roles played by different objects are expressed by the rule h. For example, in the learning stage in generation-t, the model sees three data pairs $\mathbf{d}^{t-1}$: (`[B,C],undetermined`), (`[B],off`), and (`[A,B,C],on`). We then expect the model to guess a rule like $h^t =$ `{A:on, B:off, C:undetermined}`, which means `A` can trigger the light to be `on`, `B` cannot, and `C` is not sure. In the sampling stage, we will feed the above h^t together with the instructions to the model and hope it generates more examples following $p(\mathbf{d} \mid h^t)$. Hence the model's output might be (`[A,C],on`), (`[A,B],on`), and (`[C],undetermined`). Treating the above examples as $\mathbf{d}^t$, the model in the next generation can induce the corresponding h^{t+1} by selecting the hypothesis with the largest of $P_{lmw}(h \mid \mathbf{d}^t)$. To ensure the generalizability of our analysis, we conduct experiments on `GPT3.5`, `GPT4`, `Claude3-haiku`, and `Mixtral-8x7b`. Please refer to Figure 2 and Appendix D.3 for more details.

5.1 How the Knowledge of LLM Agents Evolves

Convergence of the posterior. We start from the guarantees mentioned in Proposition 1 under an *imitation-only* setting. In this experiment, we choose $M = 5$ to better illustrate the posterior distribution $P_{lmw}(h)$ (there are $3^5 = 243$ possible h). Thanks to the instruction-following ability, all LLMs we considered always return rules in the correct format, where the probabilities of all format-related tokens are almost one. We can then calculate $P_{lmw}(h)$ or $P_{lmw}(h \mid \mathbf{d})$ by multiplying the probabilities of specific tokens in their response (see Appendix D.1 for more details).

We first demonstrate the convergence of $P_{lmw}(h)$, i.e., $P_{lmw}(h) \to \mathbb{1}(\cdot)$, which can be supported by the decreasing of the posterior's entropy, i.e., $H(P_{lmw}(h))$. As illustrated in the first panel in Figure 3, $H(P_{lmw}(h))$ gradually decreases to almost zero as iterated learning goes on, which verifies our theory that $P_{lmw}(h)$ will converge to a one-hot-like distribution[1]. Smaller temperature τ makes the convergence faster, which matches our intuitions as well. To better illustrate how different h evolves during iterated learning, similar to what we did for the compositional language experiment in Appendix C.2, we also provide the probability of all possible $h \in \mathcal{H}$ in a similar fashion. Note that for this problem, it is impossible to get the prior distribution $P_0(h)$, because we must give the model at least one example as $\mathbf{d}^0$. So in the rightmost two panels in Figure 3, we compare the posterior of the first and the sixth generations and see that the posterior becomes sparser.

[1]Note that the entropy of a uniformly distributed h and a one-hot h are roughly 5.34 and 0, respectively.

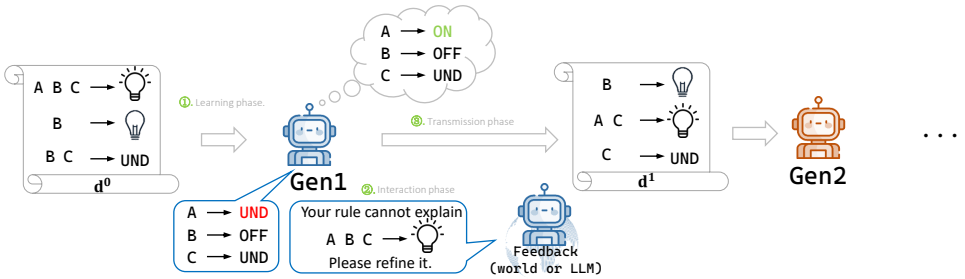

Figure 2: Demonstration of conducting iterated ICL on the ACRE task.

Converged h under different likelihood and priors. We then show how iterated learning amplifies specific biases implied in the prior, i.e., $h^{T*} \rightarrow \text{argmax}_h P_0(h)$, and how the bias and likelihood influence the converging behavior. Note that $P_0(h)$ represents LLM's belief given the instruction prompt $\mathbf{w}$, where the few-shot examples are not included. Thanks to the phenomenon mentioned in (McCoy et al. 2023), where the confidence of LLM's prediction is heavily influenced by its degree of familiarity with the output phrases, we can manipulate the prompt to create spurious correlations and hence implicitly control bias in $P_0(h)$[2]. Specifically, we change the name of the last object from "`E`" to "`screen`" and add a sentence like "`Turn off the screen after the experiment.`" in the instruction prompt. Then all h with `screen:off` would have higher prior under this prompt. We use six different prompts to introduce different levels of biases (see Appendix D.2 for more details).

We then control the strength of the likelihood by selecting different h^*, i.e., the ground truth rule we want to recover. For the strong likelihood case, we select h^* where four objects are being `on` while there is only one in the weak likelihood case. The status of `screen` in both cases is `undetermined`. Due to the nature of the ACRE task, i.e., the existence of an `on`-object in the input will trigger the light on, there might be more examples whose outputs are `on` when the likelihood is strong. Then it is harder for the model to amplify the prior bias that favors the status of `screen` to be `off`. Because the likelihood and prior compete with each other during iterated learning, as illustrated by Equation (1).

This competing relationship can be well depicted by the middle two panels in Figure 3, where we track the probability of $P_{lmw}(\texttt{screen:off})$ at the end of each generation. The converging speed under different settings correlates with the level of prior bias well. Furthermore, we find it is easier for the bias to be amplified when the likelihood is weaker, as five out of six curves converge to one in the right panel. This trend is more clear in Figure 12 and 13, where curves with the same level of bias are shown together. These results give us a good picture of how the likelihood and prior bias interact with each other during evolution and also verify the correctness of the Bayesian-IL framework for LLM agents. Plus, we plot the histograms of $P_{lmw}(h)$ under weak-likelihood-high-bias case in the rightmost two panels in Figure 3, which also demonstrates the amplified bias (the blue region grows).

Table 1: Results when adding different interaction phases. Column "BOTH" represents the ratio of converged h^T who correctly predict all 8 examples in $\mathbf{d}^0$ and have `screen:off` (i.e., r_{20}=`off`). The Mixtral model does not have self-refine results, as it violates the instructions too much.

	Old GPT3.5-Turbo 1106			New GPT3.5-Turbo 0125			Claude3-haiku-20240307			Mixtral-8x7b		
	Corr. $\mathbf{d}^0$	r_{20} = `off`	BOTH	Corr. $\mathbf{d}^0$	r_{20} = `off`	BOTH	Corr. $\mathbf{d}^0$	r_{20} = `off`	BOTH	Corr. $\mathbf{d}^0$	r_{20} = `off`	BOTH
imitation-only	4.8±1.56	70%	15%	5.6±1.11	80%	0%	6.4±1.50	90%	30%	5.5±2.01	20%	10%
w. self-refine	7.0±0.60	40%	20%	6.6±1.11	95%	35%	7.0±0.70	60%	15%	-	-	-
w. hypo-search	7.7±0.21	80%	45%	7.4±0.66	100%	55%	7.5±0.67	90%	50%	6.5±1.97	30%	30%

Influence of the interaction phase and $\mathcal{H}_{\text{eff}}$. Finally, we introduce the interaction phase and show that $h^{T*} \rightarrow \text{argmax}_{h \in \mathcal{H}_{\text{eff}}} P_0(h)$. Two mechanisms are considered in this experiment: self-refine (Madaan et al. 2023), where the feedback comes from the model's own response; and hypothesis-search (Qiu et al. 2024), where the feedback comes from an external ground-truth interpreter. We can consider the self-refine as using an *imperfect* $\mathcal{H}_{\text{eff}}$. In both settings, the LLM refines its proposed h^t at the end of each generation by checking and reporting whether this h^t can explain all samples in $\mathbf{d}^0$ (details in Appendix D.3).

In this experiment, we give the model 8 different examples in $\mathbf{d}^0$, where all these examples can be explained by both h^* and $\hat{h}$. We first select an $\hat{h}$ from all 162 candidates ($3^4 \times 2$) and then create h^* by changing the value of `screen` to `off`. Under this setting, both h^* and $\hat{h}$ belong to $\mathcal{H}_{\text{eff}}$ (i.e., mappings

[2]This phenomenon also inspires us to manipulate the prompt $\mathbf{w}$ to inject useful bias in LLM's evolution.

https://doi.org/10.52202/079017-1220

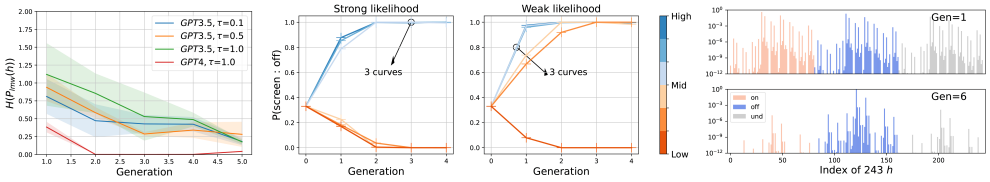

Figure 3: Left: the mean and standard deviations of $H(P_{lmw}(h))$ of experiments with different h^* and $\mathbf{d}^0$ (5 different seeds). Middle two: the probability of `screen` being `off`, where different colors represent six different levels of spurious bias. Right: the histogram of all $P_{lmw}(h)$ in the first and sixth generation, where the bars are colored based on the value of the last object in h.

with perfect training accuracy in $\mathbf{d}^0$) and h^* is what we want our model to converge to. See Table 1, where we run experiments under 10 different h^* and report three quantitative metrics of the last iteration, i.e., h^{T*}. We first report the number of correct predictions (mean and standard error) in $\mathbf{d}^0$, which demonstrates how well the method constrains $h^{T*} \in \mathcal{H}_{\text{eff}}$. The imitation-only method performs the worst, which warns us if the LLM keeps learning from the corpus generated by the agents in previous generations without any evaluation or filtering, even the training accuracy on given $\mathbf{d}^0$ would be harmed. Because hallucination or incorrectness can aggregate through generations. Adding the interaction phase can mitigate this problem efficiently, which is why most of the related works contain a "data-selection" or "data-reranking" phase. The fact that the hypo-search outperforms self-refine indicates the importance of an appropriate $\mathcal{H}_{\text{eff}}$, which means a good reward (or evaluating) model is crucial for these iterated training methods. Another metric is the ratio of h^{T*} with `screen:off`, which measures how well the bias is amplified (we here assume this bias is beneficial and wish it to be amplified, as the compositionality bias in emergent communication example showed in Appendix C.2). We find all these methods can amplify the bias to some extent and hypo-search also performs the best. Last, combined with the requirement of good training accuracy and amplifying bias, we report the ratio that the algorithm successfully chose $h^{T*} = h^*$. As illustrated in the last column of the table, adding an interaction phase with good $\mathcal{H}_{\text{eff}}$ always brings benefits.

In summary, this section verifies the correctness of the proposed analysis in LLM agents when the hypothesis is observable. The results remind us to pay more attention to whether the bias is beneficial or not and to design a better interaction phase as well.

6 Experimental Verifications when the Hypothesis is Implicit

Section 5 demonstrates that the Bayesian-IL framework can predict the behavior of LLM agents when h is explicitly defined and utilized when generating new examples. This section considers a hidden h scenario that is more general in most LLM systems. We start from a few-shot self-data-augmentation task, where the LLM keeps generating new examples to augment the data pool. In this process, h is implicitly selected when the few-shot examples are given, as stated in Proposition 2.

Experimental settings. We choose a scenario where on-policy self-data-augmentation is repeated for several generations. Consider using an LLM to generate multiple examples of an acronym-brainstorm task, where each example d is composed of an acronym and the corresponding word list, e.g., `Acronym:IL`; `List:["infinite","loop"]`. The h determines the properties of d. We hold a data pool $\mathcal{D}_{\text{pool}}$, which contains 20 random samples as $\mathbf{d}^0$ at the beginning of the experiment. In each generation, the model will generate 20 extra examples based on the data generated by itself in the previous round, i.e., $\mathbf{d}^t \sim P_{lmw}(\mathbf{d} \mid \mathbf{d}^{t-1})$. The generated $\mathbf{d}^t$ will be pushed into $\mathcal{D}_{\text{pool}}$, which simulates a scenario in which the available data keeps growing when we conduct self-data augmentation. In this experiment, h is hidden and might have different interpretations. We consider that h represents two types of acronyms, i.e., h_{easy}, where the acronym is a common word and h_{hard} otherwise. As the training data of the LLMs in our experiments is private, we instead use the ranking of the frequency of a word that appeared in common English corpus[3] as an approximation. We categorize a word as "easy" if its ranking is below 60,000; otherwise, we label it as "hard".

Bias in prior is amplified during IL. Many recent works observe that the LLM prefers to output more common words (i.e., those with higher frequency in the pertaining corpus) (McCoy et al. 2023;

[3]The frequency and ranking comes from Corpus of Contemporary American English (COCA) (Davies 2008).

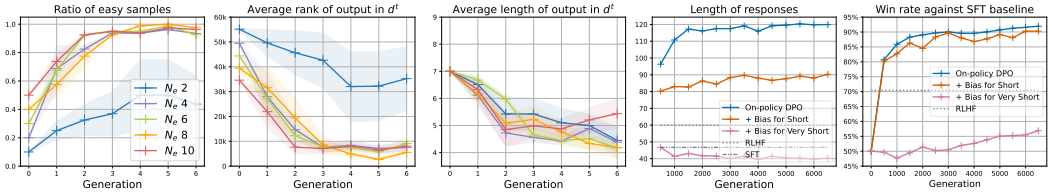

Figure 4: Leftmost three: experiments in Section 6. First: how the ratio of easy samples changes in $\mathbf{d}^t$. N_e is the number of easy examples in $\mathbf{d}^0$. Second: how the average ranking of acronyms changes. Third: how the average length of acronyms changes. Rightmost two: results of on-policy DPO in Section 7. Fourth: average length of the responses. Fifth: win rate against the SFT baseline.

Z. Wu et al. 2023), which can be considered as a bias towards h_{easy}, i.e., $P_0(h_{\text{easy}}) > P_0(h_{\text{hard}})$. Since h is hidden and we cannot directly observe it like in the previous experiment, we instead track three quantities: 1.) the proportion of easy samples in all 20 samples for each $\mathbf{d}^t$; 2.) the average ranking of $\mathbf{d}^t$, where all hard examples are ranked 60,001; and 3.) the average length of the acronyms for $\mathbf{d}^t$. As in the leftmost three panels Figure 4, the aforementioned bias is gradually amplified during iterated learning whatever the initial proportion of the easy samples in $\mathbf{d}^0$ is.

Interaction phase when h is hidden. As h is inaccessible, which forbid us to directly apply hypo-search or self-refine, we instead add a filter on the transmitted data across different generations, which plays a similar role as the interaction phase. Specifically, we use a sampled $\hat{\mathbf{d}} \sim \mathcal{D}_{\text{pool}}(d \mid h \in \mathcal{H}_{\text{eff}})$ to replace original $\mathbf{d}^{t-1}$ in the "imitation-only" setting. Based on how we sample $\hat{\mathbf{d}}$, different constraints on $\mathcal{H}_{\text{eff}}$ are implicitly imposed. We compare the behavior of five different settings, they are: 1.) $\mathcal{H}_{random}$, where $\hat{\mathbf{d}}$ is randomly sampled from $\mathcal{D}_{\text{pool}}$; 2.) $\mathcal{H}_{hard}$ where only hard examples can be sampled; 3.) $\mathcal{H}_{easy}$, opposite to the hard setting; 4.) $\mathcal{H}_{easylong}$, where the easy acronyms with longer lengths are more likely to be sampled; 5.) $\mathcal{H}_{easyshort}$, opposite to the easy-long setting.

See the first several columns of Table 2 that show the ratio of easy examples in $\mathbf{d}^t$. Compared with the random setting, all methods expect $\mathcal{H}_{\text{hard}}$ finally converges to $\mathbf{d}^t$ with more easy examples, which means the bias towards easier acronyms would be amplified when $\mathcal{H}_{\text{eff}}$ doesn't impede it. On the contrary, using $\mathcal{H}_{\text{hard}}$ successfully restrain this bias, as the average number of easy samples in $\mathbf{d}^t$ is even lower than that in $\mathbf{d}^0$. We can also design composite $\mathcal{H}_{\text{eff}}$ by choosing two properties of the data. For example, $\mathcal{H}_{easylong}$ restrains the samples with hard and short outputs, which is why they have more easy but long examples in their $\mathbf{d}^t$.

In summary, this experiment verifies that the Bayesian-IL framework still works when h is hidden: the bias is amplified generation by generation, implicitly imposing $\mathcal{H}_{\text{eff}}$ can still guide the evolution direction. Please also refer to Appendix E for more results and discussions.

Table 2: Results when adding different $\mathcal{H}_{\text{eff}}$. We color the highest and lowest numbers in each column. N_e is the number of easy examples in $\mathbf{d}^0$. Results under different settings are in Table 4 and 5.

	Ratio-easy					Avg-rank					Avg-length				
N_e=	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
Random	0.913±0.01	0.600±0.08	0.963±0.00	0.887±0.03	0.825±0.06	13519	27269	7487	10425	15871	**5.425±1.04**	4.825±0.33	5.600±1.55	5.014±1.50	4.713±0.63
Imitation-only	0.438±0.20	0.935±0.01	0.925±0.00	0.975±0.00	0.963±0.00	35235	7497	9081	5549	8075	4.450±0.86	4.387±1.40	**4.175±0.13**	4.188±0.65	5.438±1.24
Hard	**0.219±0.19**	**0.250±0.43**	**0.450±0.43**	**0.338±0.16**	**0.500±0.23**	**49869**	**46436**	**37288**	**41255**	**31903**	4.630±1.54	5.788±1.39	4.675±0.40	4.388±0.60	5.200±0.42
Easy	0.763±0.17	**1.000±0.00**	**0.988±0.00**	**1.000±0.00**	0.990±0.00	15910	**3156**	**2383**	**2924**	**2650**	**3.925±0.33**	5.263±0.06	4.713±0.06	4.240±0.08	4.893±0.71
Easylong	0.988±0.00	0.975±0.00	**0.988±0.00**	0.988±0.00	**1.000±0.00**	7063	9413	8649	6898	7404	5.209±0.41	**5.888±0.52**	**6.838±1.10**	**6.979±1.57**	**7.695±1.70**
Easyshort	**1.000±0.00**	**1.000±0.00**	0.975±0.00	**1.000±0.00**	0.988±0.00	**5671**	4223	5733	4502	5251	3.975±0.50	**4.012±1.03**	4.374±0.50	**3.950±0.03**	**4.250±0.24**

7 Experiments on In-Weights Learning: On-Policy DPO as an Example

Besides the manually designed experiments in the previous two sections, here we verify our analysis in a real preference-tuning task using on-policy DPO (Guo et al. 2024). In each round of the training, the model first samples multiple responses given the prompt (similar to the sampling stage in IL). Then, these responses are evaluated and ranked by another LLM annotator based on their level of helpfulness (the interaction phase in IL). Finally, we select the highest (lowest) ranked samples as the chosen (rejected) response and use a standard DPO algorithm (Rafailov et al. 2024) to train the policy network (the imitation phase in IL). As described before, each update of the on-policy DPO algorithm can be considered as one generation in iterated learning, because the model keeps updating its parameters using the responses generated by itself. Ranking the responses based on helpfulness is equivalent to imposing a $\mathcal{H}_{\text{helpful}}$. As a result, the phenomenon of bias amplification, the guiding

https://doi.org/10.52202/079017-1220

effect of the interaction phase design, and the influence of spurious correlation, should still hold in this practical setting.

To verify our analysis, we finetune a pretrained `llama-2-7B` model (Touvron et al. 2023) using `Antropic-HH` dataset (Bai et al. 2022) following the on-policy DPO recipe provided in (Guo et al. 2024). We study the length bias demonstrated in (Dubois et al. 2024), which means the LLM tends to prefer longer responses when answering questions. We first show that such a bias will be significantly amplified by a multi-generation self-improvement method (on-policy DPO) compared with a non-self-iterated method (RLHF, (Ouyang et al. 2022)). As demonstrated by the blue curve and the dotted line of the fourth panel in Figure 4, the average length of the responses from the model trained using on-policy DPO is much larger than the SFT baseline and RLHF counterparts. With the increase of the win rate against SFT[4], the average response length also keeps increasing. To restrain this bias, we impose $\mathcal{H}_{\text{short}}$ by adding a sentence like "you are a laconic agent and prefer concise answers" to the annotator LLM, just like how we manipulate the spurious correlation between `screen` and `off` in Section 5. Then, combining with the existing interaction phase that requires $h \in \mathcal{H}_{\text{helpful}}$, this design is equivalently imposing a constraint of $h \in \mathcal{H}_{\text{helpful}} \cap \mathcal{H}_{\text{short}}$. Hence as illustrated by the orange curves in the last two panels, the on-policy DPO can then generate shorter responses (the increasing speed is also restrained) while keeping a high level of helpfulness. However, if our constraints of the length are too strong, which makes $\mathcal{H}_{\text{helpful}} \cap \mathcal{H}_{\text{veryshort}} = \Phi$, the model's helpfulness will then be significantly harmed, as demonstrated by the pink curves in these two panels.

In summary, we find all our analysis on the Bayesian-IL still holds for a practical preference-tuning task: the biases would be amplified and a suitable interaction phase can control it as long as we can figure out them. However, some biases are inevitably hidden and are also amplified during LLM's evolution. Hence how to pinpoint these biases, or finding a method that can restrain malicious biases even without explicitly knowing them, would be interesting directions to explore in the future.

8 Conclusion

This paper examines the potential and ongoing evolutions of LLM agents by drawing parallels with human cultural evolution, where the latter is a well-established subject in cognitive science. By demonstrating that the sampling and learning procedures of LLMs in various algorithms can be effectively approximated by Bayesian inference, we successfully apply the Bayesian-IL framework to elucidate and steer the evolution of LLM agents. The presented theory and accompanying experiments not only provide deeper insights into LLM behavior from a top-down perspective but also hold the potential to inspire the design of more efficient self-evolution algorithms.

Acknowledgments and Disclosure of Funding

This work was supported in part by the Natural Sciences and Engineering Resource Council of Canada, the Fonds de Recherche du Québec - Nature et technologies (under grant ALLRP-57708-2022), the Canada CIFAR AI Chairs program, the BC DRI Group, Calcul Québec, Compute Ontario, and the Digital Resource Alliance of Canada

References

Walter R Gilks, Sylvia Richardson, and David Spiegelhalter (1995). *Markov chain Monte Carlo in practice*. CRC Press.

Søren Feodor Nielsen (2000). "The stochastic EM algorithm: estimation and asymptotic results." *Bernoulli*, pages 457–489.

Aad W Van der Vaart (2000). *Asymptotic statistics*. Volume 3. Cambridge university press.

Simon Kirby, Mike Dowman, and Thomas L Griffiths (2007). "Innateness and culture in the evolution of language." *Proceedings of the National Academy of Sciences* 104.12, pages 5241–5245.

Mark Davies (2008). "The Corpus of Contemporary American English." URL: www.english-corpora.org/coca/.

[4]We use `GPT4` to compare the helpfulness between responses generated by different models, as did Rafailov et al. (2024).

Thomas L Griffiths, Brian R Christian, and Michael L Kalish (2008). "Using category structures to test iterated learning as a method for identifying inductive biases." *Cognitive Science* 32.1, pages 68–107.
Simon Kirby, Hannah Cornish, and Kenny Smith (2008). "Cumulative cultural evolution in the laboratory: An experimental approach to the origins of structure in human language." *PNAS* 105.31, pages 10681–10686.
David Lewis (2008). *Convention: A philosophical study*. John Wiley & Sons.
Aaron Beppu and Thomas Griffiths (2009). "Iterated learning and the cultural ratchet." *Proceedings of the Annual Meeting of the Cognitive Science Society*. Volume 31.
Florencia Reali and Thomas L Griffiths (2009). "The evolution of frequency distributions: Relating regularization to inductive biases through iterated learning." *Cognition* 111.3, pages 317–328.
Nicolas Fay, Simon Garrod, Leo Roberts, and Nik Swoboda (2010). "The interactive evolution of human communication systems." *Cognitive science* 34.3, pages 351–386.
Amy Perfors, Joshua B Tenenbaum, Thomas L Griffiths, and Fei Xu (2011). "A tutorial introduction to Bayesian models of cognitive development." *Cognition* 120.3, pages 302–321.
Elizabeth Bonawitz, Stephanie Denison, Thomas L Griffiths, and Alison Gopnik (2014). "Probabilistic models, learning algorithms, and response variability: sampling in cognitive development." *Trends in cognitive sciences* 18.10, pages 497–500.
Vanessa Ferdinand, Simon Kirby, and Kenny Smith (2014). "Regularization in language evolution: On the joint contribution of domain-specific biases and domain-general frequency learning." *Evolution of Language: Proceedings of the 10th International Conference (EVOLANG10)*. World Scientific, pages 435–436.
Simon Kirby, Monica Tamariz, Hannah Cornish, and Kenny Smith (2015). "Compression and communication in the cultural evolution of linguistic structure." *Cognition* 141, pages 87–102.
Danielle J Navarro, Andrew Perfors, Arthur Kary, Scott D Brown, and Chris Donkin (2018). "When extremists win: Cultural transmission via iterated learning when populations are heterogeneous." *Cognitive Science* 42.7, pages 2108–2149.
Vanessa Ferdinand, Simon Kirby, and Kenny Smith (2019). "The cognitive roots of regularization in language." *Cognition* 184, pages 53–68.
Shangmin Guo, Yi Ren, Serhii Havrylov, Stella Frank, Ivan Titov, and Kenny Smith (2019). "The emergence of compositional languages for numeric concepts through iterated learning in neural agents." *The 3rd workshop on Emergent Communication, NeurIPS*.
Yasamin Motamedi, Marieke Schouwstra, Kenny Smith, Jennifer Culbertson, and Simon Kirby (2019). "Evolving artificial sign languages in the lab: From improvised gesture to systematic sign." *Cognition* 192, page 103964.
Yuchen Lu, Soumye Singhal, Florian Strub, Aaron Courville, and Olivier Pietquin (2020). "Countering language drift with seeded iterated learning." *International Conference on Machine Learning*. PMLR, pages 6437–6447.
Hossein Mobahi, Mehrdad Farajtabar, and Peter Bartlett (2020). "Self-distillation amplifies regularization in Hilbert space." *Advances in Neural Information Processing Systems* 33, pages 3351–3361.
Yi Ren, Shangmin Guo, Matthieu Labeau, Shay B. Cohen, and Simon Kirby (2020). "Compositional languages emerge in a neural iterated learning model." *International Conference on Learning Representations*.
Ankit Vani, Max Schwarzer, Yuchen Lu, Eeshan Dhekane, and Aaron Courville (2021). "Iterated learning for emergent systematicity in vqa." *International Conference on Learning Representations*.
Chi Zhang, Baoxiong Jia, Mark Edmonds, Song-Chun Zhu, and Yixin Zhu (2021). "Acre: Abstract causal reasoning beyond covariation." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10643–10653.
Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. (2022). "Training a helpful and harmless assistant with reinforcement learning from human feedback." arXiv: 2204.05862.
Yasamin Motamedi, Lucie Wolters, Danielle Naegeli, Simon Kirby, and Marieke Schouwstra (2022). "From improvisation to learning: How naturalness and systematicity shape language evolution." *Cognition* 228, page 105206.
Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. (2022). "Training language models to follow instructions with human feedback." *Advances in Neural Information Processing Systems* 35, pages 27730–27744.

 https://doi.org/10.52202/079017-1220

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi (2022). "Self-instruct: Aligning language model with self generated instructions." arXiv: 2212.10560.
Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma (2022). "An Explanation of In-context Learning as Implicit Bayesian Inference." *International Conference on Learning Representations*.
Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. (2023). "Weak-to-strong generalization: Eliciting strong capabilities with weak supervision." arXiv: 2312.09390.
Emil Carlsson, Devdatt Dubhashi, and Terry Regier (2023). "Iterated learning and communication jointly explain efficient color naming systems." arXiv: 2305.10154.
Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang (2023). "Raft: Reward ranked finetuning for generative foundation model alignment." arXiv: 2304.06767.
Thomas L Griffiths, Jian-Qiao Zhu, Erin Grant, and R Thomas McCoy (2023). "Bayes in the age of intelligent machines." arXiv: 2311.10206.
Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. (2023). "Reinforced self-training (rest) for language modeling." arXiv: 2308.08998.
Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. (2023). "Self-refine: Iterative refinement with self-feedback." arXiv: 2303.17651.
R Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, and Thomas L Griffiths (2023). "Embers of autoregression: Understanding large language models through the problem they are trained to solve." arXiv: 2309.13638.
Yi Ren, Samuel Lavoie, Mikhail Galkin, Danica J. Sutherland, and Aaron Courville (2023). "Improving Systematic Generalization using Iterated Learning and Simplicial Embeddings." *Thirty-seventh Conference on Neural Information Processing Systems*.
Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson (2023). "Model Dementia: Generated Data Makes Models Forget." arXiv: 2305.17493.
Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. (2023). "Llama 2: Open foundation and fine-tuned chat models." arXiv: 2307.09288.
Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim (2023). "Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks." arXiv: 2307.02477.
Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang (2023). "Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint." *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*.
Wei Xiong, Hanze Dong, Chenlu Ye, Han Zhong, Nan Jiang, and Tong Zhang (2023). "Gibbs sampling from human feedback: A provable kl-constrained framework for rlhf." arXiv: 2312.11456.
Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley (2023). "Baize: An open-source chat model with parameter-efficient tuning on self-chat data." arXiv: 2304.01196.
Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu (2024). "Self-play fine-tuning converts weak language models to strong language models." arXiv: 2401.01335.
Zheng Chenhao, Zhang Jieyu, Kembhavi Aniruddha, and Krishna Ranjay (2024). "Iterated Learning Improves Compositionality in Large Vision-Language Models." arXiv: 2402.04792.
Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto (2024). "Length-controlled alpacaeval: A simple way to debias automatic evaluators." arXiv: 2404.04475.
Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, et al. (2024). "Direct language model alignment from online ai feedback." arXiv: 2402.04792.
Linlu Qiu, Liwei Jiang, Ximing Lu, Melanie Sclar, Valentina Pyatkin, Chandra Bhagavatula, Bailin Wang, Yoon Kim, Yejin Choi, Nouha Dziri, et al. (2024). "Phenomenal Yet Puzzling: Testing Inductive Reasoning Capabilities of Language Models with Hypothesis Refinement." *International Conference on Learning Representations*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn (2024). "Direct preference optimization: Your language model is secretly a reward model." *Advances in Neural Information Processing Systems* 36.

Yi Ren and Danica J Sutherland (2024). "Learning Dynamics of LLM Finetuning." arXiv: 2407.10490.

Yuan Weizhe, Pang Richard Yuanzhe, Cho Kyunghyun, Sukhbaatar Sainbayar, Xu Jing, and Weston Jason (2024). "Self-Rewarding Language Models." *arXiv preprint arXiv:2401.10020*.

Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, and William Yang Wang (2024). "Perils of Self-Feedback: Self-Bias Amplifies in Large Language Models." arXiv: 2402.11436.

 https://doi.org/10.52202/079017-1220

A Discussions of the Proposed Theory and its Applicability to Real Methods

A.1 Assumptions of Bayesian-IL and practical Scenarios

As typical in theoretical machine learning research, some assumptions are needed to prove results about models' behavior; these assumptions are often not *exactly* satisfied by practical algorithms. So, we elaborate here on the important assumptions we made and when practical algorithms break them.

1. Assumptions for the theoretical analysis. To derive the guarantees of Proposition 1, we first model the interaction phase as a binary filter on $h \in \mathcal{H}_{\text{eff}}$ and also assume a shared prior $P_0(h)$ among all agents involved in Bayesian-IL. We also model the LLM's in-context behavior as a Bayesian agent and assume the number of samples during the imitation phase is sufficient.

2. Assumptions we can break for iterative ICL experiments.

- Binary filter on $h \in \mathcal{H}_{\text{eff}}$. All our LLM experiments break this assumption (the pure Bayesian example in Appendix C.2 does not). For example, in the ACRE experiments, we use self-refine and hypothesis search as the interaction phase. Self-refine asks the model to evaluate the responses, and the hypothesis search uses an external interpreter: they both manipulate h by feeding messages to the LLM, rather than a binary filter. (When using an external interpreter, h is usually filtered before forming the refinement feedback.) For the experiments in Section 6 and 7, where h is implicit, we re-rank all the generated samples in $\mathcal{D}_{\text{pool}}$ and take a weighted sample during imitation, similar to re-ranking the generated examples in ReST. Since all these interaction designs are commonly applied in the community, and our theory describes their qualitative behaviors well despite strictly violating the assumption, we believe our methods can shed more light on other practical methods with similar designs, like self-reward (Weizhe et al. 2024), iterative-DPO (Xiong, Dong, Ye, Z. Wang, et al. 2023), etc.
- Identical $P_0(h)$ for agents in different generations. Although this assumption makes it easier to derive Proposition 1, slightly relaxing it will not change the whole story: we only require different agents to share a similar tendency towards a specific bias. To verify this, we conduct several experiments when the agents in different generations are different LLMs (e.g., `GPT3.5` plays with `Claude3` in Appendix D and E). The phenomena claimed by the theory still hold.
- The Bayesian learning assumption, i.e., $P_{lm}(h) = P(h \mid \mathbf{d}) \propto p(\mathbf{d} \mid h)P_0(h)$. Although this assumption is necessary for drawing a parallel between iterated learning and the EM algorithm and hence getting a guarantee for the amplified bias, the practical in-weights learning (IWL for short) method usually does not strictly follow this assumption, because people usually early stop the training before the model perfectly learns all $\mathbf{d}^t$. However, results in Section 7 match our analysis well, which means the iterated IWL can also be depicted by iterated learning to some extent. That is because although there are plenty of finetuning methods with different targets or loss functions, their aims are consistent: increasing the likelihood of $p(\mathbf{d}_{\text{train}} \mid h)$ under instructions, which aligns with Bayesian targets well. Furthermore, we find the increased bias or decreased creativity during iterated finetuning has also been extensively mentioned in many related works (Touvron et al. 2023; W. Xu et al. 2024), which also supports our analysis.

A.2 Why we Start from Two "Toyish" Tasks

The experimental settings in Section 5 and 6 are relatively manual and toyish. The main reason for us to start from them is that we want to *directly observe* some quantitive numbers described by the theory, which we believe would provide stronger support for the analysis. For the explicit h case, we chose the ACRE task because of its simple $\mathcal{H}$, making it possible to observe the distribution and entropy of all possible h. We believe observing the model's logits supports our theory more directly than merely observing the accuracy or other quantitative metrics.

For the implicit h case, we chose the acronym task, which is a prototype of self-data-augmentation in self-instruct. We initially consider the conditional creative writing task (quite common in many related works), where the model needs to write a passage (i.e., the list in our settings) based on several topic words (i.e., the acronym). However, constrained by the context length of LLMs,

we can't generate more than 4 examples in one generation, which makes it hard to calculate the statistics of $\mathbf{d}^t$. Remember the model will generate 20 extra $\mathbf{d}^t$ based on 20 $\mathbf{d}^{t-1}$ in our acronym experiments. In summary, although the experiments studied in our paper look artificial, they are reasonable approximations of real tasks.

Last, in a concurrent work W. Xu et al. (2024), the authors study practical applications like machine translation, creative writing, math reasoning, etc, in an iterative ICL setting. Their observations match our theoretical analysis quite well: bias is amplified generation by generation, and introducing external feedback can mitigate it. However, due to the complexity of the tasks they considered, they can only observe the average bias and the skew level using several conclusive quantitative metrics. Hence we believe that by combining our theoretical analysis, detailed observations on artificial examples, and the evidence from real applications in W. Xu et al. (2024), one can draw a good overview of how LLM would evolve in an iterated ICL setting.

A.3 How our analysis brings benefits to practical algorithms

Besides the method proposed in Section 7, where we manipulate the instructions prompt of the annotator LLM during the interaction phase, our experiments and analysis also provide the following potential approaches to guide the model's evolution:

- Select $\mathbf{d}^0$ that makes more *confident and correct* predictions on the target task. Manually selecting good in-context examples is intuitive. Our analysis, though, suggests taking the model's confidence (i.e., the logits) into account, because the theory claims that the likelihood and bias in prior are competing with each other during evolution. From Figure 3, we see the model evolves faster if the likelihood of $\mathbf{d}^0 \mid h^*$ is weaker. The results in Figure 4 also provide similar insights: the related biases are amplified slower when the number of easy samples in $\mathbf{d}^0$ decreases.
- Designing a good interaction phase is important: more accurate $\mathcal{H}_{\text{eff}}$ leads to better performance. This can be supported by comparing self-refine and hypothesis-search in our paper. The paper W. Xu et al. (2024) also claims that external feedback with more accurate assessments or feedback from a larger model can reduce the amplified bias.
- Manipulating the instruction prompt: in our analysis, both $P_0(h)$ and $P_{lm}(h)$ are the model's predictions conditioned on the instruction prompt $\mathbf{w}$. Hence adding preference in the task instruction (or changing the system prompt) during evolution could be an effective way of guiding the model's evolution. Our ACRE experiments show the feasibility of this: remember we can introduce spurious correlation by adding one sentence to the instruction. Hence it is also possible to guide the model's evolution by feeding appropriate prompts during learning and sampling.
- Manipulating the temperature: Bayesian-IL theory studies the evolution of the distribution, so the temperature should also be an important factor for the evolution, as illustrated in Figure 3. We left the exploration between temperature and different phases in IL in the future.

B Proofs related to Bayesian Agents

B.1 Recap the Proof of Expectation-Maximization Algorithm

To get a clear picture of the asymptotic behavior of imitation-only iterated learning, we first recap how a typical expectation-maximization (EM) algorithm converges when the target function is posterior distribution[5]. Consider a statistical model that generates a set of observable data samples $\{\mathbf{x}_i\}_{i=1}^m$ and the corresponding hidden variables $\{\mathbf{z}_i\}_{i=1}^m$. The generating mechanism can be expressed as $P(\mathbf{x} \mid \mathbf{z}, \theta)$, where θ is a set of unknown parameters determining this distribution. To get a

[5]The MLE (maximum likelihood estimation) version of the EM algorithm is more common in textbooks.

https://doi.org/10.52202/079017-1220

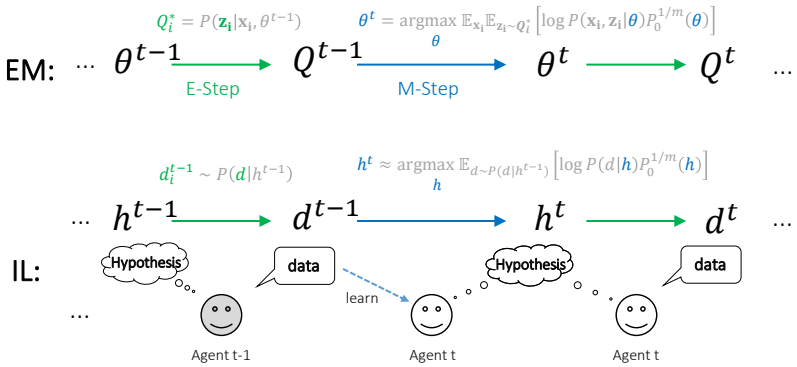

Figure 5: Illustrations of typical EM algorithm and an imitation-only iterated learning method.

MAP(maximum a posterior) estimation of θ, we need to optimize the following target function:

$$\begin{aligned}\mathcal{L}(\theta) &= \log P(\theta \mid \mathbf{x}_1, \ldots, \mathbf{x}_m) \\ &= \log P_0(\theta) + \log P(\mathbf{x}_1, \ldots, \mathbf{x}_m \mid \theta) - \log P(\mathbf{x}_1, \ldots, \mathbf{x}_m) \\ &= \sum_{i=1}^{m} \frac{1}{m} \log P_0(\theta) + \sum_{i=1}^{m} \log P(\mathbf{x}_i \mid \theta) - \log P(\mathbf{x}_1, \ldots, \mathbf{x}_m) \\ &= \sum_{i=1}^{m} \log \left(P(\mathbf{x}_i \mid \theta) P_0^{\frac{1}{m}}(\theta) \right) - \log P(\mathbf{x}_1, \ldots, \mathbf{x}_m) \end{aligned} \tag{4}$$

where $P_0(\theta)$ is the prior distribution of parameters. As the marginal distribution $P(\mathbf{x}_i \mid \theta)$ is hard to calculate due to the existence of the hidden variable $\mathbf{z}_i$, our target function can then be expressed as

$$\tilde{\mathcal{L}}(\theta) = \mathcal{L}(\theta) + \text{const} = \sum_{i=1}^{m} \log \left(\sum_{\mathbf{z}_i} P(\mathbf{x}_i, \mathbf{z}_i \mid \theta) P_0^{\frac{1}{m}}(\theta) \right) \tag{5}$$

where the term $\log P(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ is eliminated as it doesn't depend on θ. The target function above is still hard to tackle due to the summation inside the logarithmic function. To solve this, we first introduce an auxiliary function $Q_i(\mathbf{z}_i)$, which is a probability distribution over $\mathbf{z}_i$, and reformulate the target as

$$\tilde{\mathcal{L}}(\theta) = \sum_{i=1}^{m} \log \left(\sum_{\mathbf{z}_i} Q_i(\mathbf{z}_i) \frac{P(\mathbf{x}_i, \mathbf{z}_i \mid \theta) P_0^{\frac{1}{m}}(\theta)}{Q_i(\mathbf{z}_i)} \right) = \sum_{i=1}^{m} \log \left(\mathbb{E}_{\mathbf{z}_i \sim Q_i} \left[\frac{P(\mathbf{x}_i, \mathbf{z}_i \mid \theta) P_0^{\frac{1}{m}}(\theta)}{Q_i(\mathbf{z}_i)} \right] \right). \tag{6}$$

Given the concavity of the logarithmic function, we can use Jensen's inequality to get a lower bound of $\tilde{\mathcal{L}}(\theta)$:

$$\tilde{\mathcal{L}}(\theta) \geq \mathcal{J}(\theta, Q) = \sum_{i=1}^{m} \mathbb{E}_{\mathbf{z}_i \sim Q_i} \left[\log \frac{P(\mathbf{x}_i, \mathbf{z}_i \mid \theta) P_0^{\frac{1}{m}}(\theta)}{Q_i(\mathbf{z}_i)} \right]. \tag{7}$$

The EM algorithm then maximizes this lower bound by alternatively optimizing θ and Q_i for several rounds.

In the **E-step**, as illustrated in Figure 5, we use the estimated θ^{t-1} in the previous round to find $Q_i^* = \text{argmax}_Q \mathcal{J}(\theta^{t-1}, Q)$. Specifically, we need to push this lower bound to be tight by making the equality in Equation (7) hold. Following the properties of Jensen's inequality, the equality only holds when $\frac{P(\mathbf{x}_i, \mathbf{z}_i \mid \theta^{t-1}) P_0^{\frac{1}{m}}(\theta)}{Q_i(\mathbf{z}_i)}$ is a constant. Combining this requirement and the fact that $\sum_{\mathbf{z}_i} Q_i(\mathbf{z}_i) = 1$, we can calculate the optimal $Q_i^*(\mathbf{z}_i) = P(\mathbf{z}_i \mid \mathbf{x}_i, \theta^{t-1})$, which is the posterior distribution of $\mathbf{z}_i$ given the observable data $\mathbf{x}_i$ and the fixed parameters θ^{t-1}.

https://doi.org/10.52202/079017-1220

In the **M-step**, we plug in the estimated Q_i^* to $\mathcal{J}(\theta, Q)$ to get the target function as:

$$\mathcal{J}(\theta; Q^*) = \sum_{i=1}^{m} \mathbb{E}_{\mathbf{z}_i \sim Q_i^*} \left[\log \frac{P(\mathbf{x}_i, \mathbf{z}_i \mid \theta) P_0^{\frac{1}{m}}(\theta)}{Q_i^*} \right] = \sum_{i=1}^{m} \mathbb{E}_{\mathbf{z}_i \sim Q_i^*} \left[\log \left(P(\mathbf{x}_i, \mathbf{z}_i \mid \theta) P_0^{\frac{1}{m}}(\theta) \right) \right] - c, \tag{8}$$

where $c = \mathbb{E}_{\mathbf{z}_i \sim Q_i^*}[Q_i^*]$ is a constant term and can be neglected while optimizing θ. In this step, we can calculate $\theta^t = \text{argmax}_\theta \, \mathcal{J}(\theta; Q^*)$ using gradient descent or other parameter estimation methods.

In summary, the E-step ensures a tight lower bound $\mathcal{J}(\theta, Q)$ and the M-step finds better θ to make it larger. The two steps cooperate to ensure a series of estimations of θ for which $\mathcal{L}(\theta)$ is non-decreasing. Finally, the estimation of parameters will converge to the one that maximizes the posterior distribution, i.e., $\mathbb{E}(\theta^*) = \text{argmax}_\theta \, P(\theta \mid \mathbf{x}_1, \ldots, \mathbf{x}_m)$, if it is convex.

B.2 Proof: Convergence Behavior of Bayesian Agents in Iterated Learning

Proposition 1. *Consider several Bayesian agents sharing the same prior $P_0(h)$ are conducting iterated learning for T generations. If T is sufficiently large, any agent$_t$ with $t > T$ will have*

$$P_{lm}(h) \to \mathbb{1}(h = h^{T*})$$

where h^{T*} is a stationary point (e.g. a local maximum) of $P_0(h)$ subject to $h \in \mathcal{H}_{\text{eff}}$.

Proof. The proof of this proposition can be divided into two steps. In the first step, we show that imitation-only iterated learning shares similar convergence behavior with a standard EM algorithm. In the second step, we show the "selecting" pressure introduced via the interaction phase doesn't break the necessary conditions of the convergence in the first step. Merging these two steps leads to the proposition.

Step 1: imitation-only iterated learning as a special EM

Recall the imitation-only iterated learning illustrated in the bottom part in Figure 5, where the hypothesis held by the agent in the $(t-1)$-th generation is represented by h^{t-1}. With this hypothesis, the agent will generate m data samples using $P(d \mid h^{t-1})$, denoted $\mathbf{d}^{t-1} \triangleq [d_1^{t-1}, \ldots, d_m^{t-1}]$. In the t-th generation, a new agent will first update its posterior probability using $P(h|\mathbf{d}^{t-1})$, and then select h^t by picking the one with the largest posterior. As there are multiple data samples in $\mathbf{d}^{t-1}$, this process can be expressed as

$$\begin{aligned}
h^t &= \underset{h}{\text{argmax}} \log P(h \mid d_1^{t-1}, d_2^{t-1}, \ldots, d_m^{t-1}) \qquad (9) \\
&= \underset{h}{\text{argmax}} \log \left(\frac{p(d_1^{t-1}, d_2^{t-1}, \ldots, d_m^{t-1} \mid h) P_0(h)}{P(d_1^{t-1}, d_2^{t-1}, \ldots, d_m^{t-1})} \right) \\
&= \underset{h}{\text{argmax}} \log \left(P_0(h) \prod_{i=1}^{m} p(d_i^{t-1} \mid h) \right) \\
&= \underset{h}{\text{argmax}} \frac{1}{m} \log P_0(h) + \frac{1}{m} \sum_{i=1}^{m} \log p(d_i^{t-1} \mid h) \\
&\approx \underset{h}{\text{argmax}} \underset{d \sim p(d|h^{t-1})}{\mathbb{E}} [\log P_0^{\frac{1}{m}}(h)] + \underset{d \sim p(d|h^{t-1})}{\mathbb{E}} [\log p(d \mid h)] \\
&= \underset{h}{\text{argmax}} \underset{d \sim p(d|h^{t-1})}{\mathbb{E}} [\log p(d \mid h) P_0^{\frac{1}{m}}(h)]. \qquad (10)
\end{aligned}$$

Based on the analysis above, we notice that the imitation-only iterated learning and the EM algorithm are almost identical: by replacing θ and $\mathbf{z}$ to h and $\mathbf{d}$, and by removing the random variable $\mathbf{x}$[6], we can also have similar theoretical guarantees for the imitation-only iterated learning algorithm.

To prove this, we can first verify the equivalence between the imitation phase and an M-step. By comparing the target functions when calculating hidden variables (h and θ) in these two algorithms,

[6]It is unusual to apply EM with no observable data, but removing it doesn't violate any assumptions in the derivation of EM.

https://doi.org/10.52202/079017-1220

i.e., Equation (8) and (10), we can find two major differences. First, the expectation of observable samples (i.e., $\mathbb{E}_{\mathbf{x}_i}$) disappears in Equation (10), as we assume there are no "observations" in iterated learning. We can also introduce a dummy variable named $\mathbf{x}$ to the iterated learning process, and find that the existence of $\mathbf{x}$ doesn't influence the aforementioned calculation at all. Second, in IL, we can only approximate $\mathbb{E}_{d_i^{t-1}}[\cdot]$ by sampling d from $p(d \mid h^{t-1})$, while in EM, the posterior distribution $P(\mathbf{z}_i \mid \mathbf{x}_i, \theta^t)$ is usually analytically calculated. Of this discrepancy, if our $\mathbf{d}$ is a good approximation of $p(d \mid h^{t-1})$, the imitation phase in IL is equivalent to an M-step in EM.

We then verify whether the transmission phase is a good approximation of an E-step. The first thing to check is the tightness of the lower bound generated via Jensen's inequality, which guarantees the non-decreasing update of the target function across multiple generations. we can first assume the target function of the whole iterated learning process is $\mathcal{L}(h) = \log P_0(h)$, and derives its lower bound $\mathcal{J}(h; Q)$ following a similar procedure in EM:

$$
\begin{aligned}
\mathcal{L}(h) &= \log P_0(h) \\
&= m \log P_0^{\frac{1}{m}}(h) \\
&= m \log \left(\sum_{d_i} p(d_i \mid h) P_0^{\frac{1}{m}}(h) \right) \\
&= m \log \left(\sum_{d_i} Q_i(d_i) \frac{p(d_i \mid h) P_0^{\frac{1}{m}}(h)}{Q_i(d_i)} \right) \\
&= m \log \mathbb{E}_{d_i \sim Q_i} \left[\frac{p(d_i \mid h) P_0^{\frac{1}{m}}(h)}{Q_i(d_i)} \right] \\
&\geq \mathbb{E}_{d_i \sim Q_i} \log \left[\frac{p(d_i \mid h) P_0^{\frac{1}{m}}(h)}{Q_i(d_i)} \right] \triangleq \mathcal{J}(h, Q)
\end{aligned} \tag{11}
$$

The equality of Jensen's inequality holds when $\frac{p(d_i|h)P_0^{1/m}(h)}{Q_i(d_i)}$ is a constant. Using the fact that $\sum_{d_i} Q_i(d_i) = 1$, we can have the optimal $Q_i^* = p(d_i \mid h)$. In summary, as we sample each $d_i^{t-1} \sim p(d \mid h^{t-1})$, the transmission phase in IL is equivalent to an E-step in EM.

Another interesting parameter to discuss is m, i.e., the number of data samples generated by an agent in each generation. The choice of m determines how well the sampled $\mathbf{d}$ can represent the ground truth $p(d \mid h)$. For large enough m, we can prove the convergence guarantee using the above procedure. When $m = 1$, the standard EM algorithm becomes a stochastic EM approximation Gilks et al. 1995. The authors of Nielsen (2000) proved that in stochastic EM, θ in different generations form a homogeneous Markov chain whose stationary distribution over hypotheses is approximately centered on the maximum-likelihood solution. In other words, when $t > T$ for sufficiently large T, $\mathbb{E}[\theta^t]$ optimizes $\mathbb{E}_{\mathbf{x}}[P(\theta \mid \mathbf{x}_1, \ldots, \mathbf{x}_m)]$, and similarly, $\mathbb{E}[h^t]$ is the optimizer of $P_0(h)$. In other words, the dominating hypothesis in imitation-only iterated learning converges to the one with the highest prior, which is equivalent to the large m case.

Although $m = 1$ doesn't influence the converged estimation of h, a too small m will make the variance of estimation large, and hence impede the converging speed of h. Then should we choose m as large as possible? The answer is still no: too large m will also impede the converging speed. We can get some intuition by observing Equation (10), where $P_0^{1/m}(h)$ determines the effect of the prior when selecting optimal h in each generation. If m is too large, this distribution would be very flat and the preference encoded in the prior cannot influence the choice of h in this generation much – the likelihood term $P(d \mid h)$ will dominate.[7] Hence the resulting h^t would be quite close to h^{t-1}, which means the evolution of belief on h would be slow.

Actually, the choice of m is usually considered as the "bottleneck" parameter in different iterated learning algorithms. Almost all the related studies point out that the bottleneck should not be too wide

[7] A similar trend also exists in Bernstein-von Mises theorem (see e.g. Van der Vaart 2000), which claims the posterior $p(\theta \mid x_1, \ldots, x_n) = \mathcal{N}(\theta_0, n^{-1} I(\theta_0)^{-1})$, for $n \to \infty$. In other words, $P(h|d_1, \ldots, d_m)$ would become peakier as m increases, and hence h^t will be closer to h^{t-1} after the imitation phase.

or too tight (like experiments in Kirby et al. (2015) and Ren et al. (2020)). Our Bayesian analysis provides a theoretical explanation of the effect.

Step 2: the influence of introducing $\mathcal{H}_{\text{eff}}$. Let us first check the imitation phase, i.e., the E-step where h^t is calculated in Equation (9). Assume we have a perfect interaction phase that can rule our all $h \notin \mathcal{H}_{\text{eff}}$, then the target function is:

$$\begin{aligned} h^t &= \operatorname*{argmax}_{h \in \mathcal{H}_{\text{eff}}} \log P(h \mid d_1^{t-1}, d_2^{t-1}, \ldots, d_m^{t-1}) \\ &= \operatorname*{argmax}_{h} \log P(h \mid d_1^{t-1}, d_2^{t-1}, \ldots, d_m^{t-1}) \cdot \mathbb{1}(h \in \mathcal{H}_{\text{eff}}) \\ &= \operatorname*{argmax}_{h} \mathop{\mathbb{E}}_{d \sim p(d \mid h^{t-1})} \left[\log p(d \mid h) \cdot \left(P_0^{\frac{1}{m}}(h) \cdot \mathbb{1}(h \in \mathcal{H}_{\text{eff}}) \right) \right] \\ &\triangleq \operatorname*{argmax}_{h} \mathop{\mathbb{E}}_{d \sim p(d \mid h^{t-1})} \left[\log p(d \mid h) \cdot \tilde{P}_0^{\frac{1}{m}}(h) \right], \end{aligned} \tag{12}$$

where we define a "regularized" prior as $\tilde{P}_0(h) \triangleq c \cdot P_0(h) \cdot \mathbb{1}(h \in \mathcal{H}_{\text{eff}})$. Then, by substituting this prior back to $\mathcal{L}(h)$ defined in Equation (11), we find the optimal $Q_i^* = p(d_i \mid h)$ still holds. In other words, as long as we same $d_i^{t-1} \sim p(d \mid h^{t-1})$, where $h^{t-1} \in \mathcal{H}_{\text{eff}}$, all the conditions required for this proposition still hold. Furthermore, this proof also provides us an insight that adding constraints on $h \in \mathcal{H}_{\text{eff}}$ is required in both imitation and transmission phases. Hence having a powerful "data-filter" or "data-ranking" design for the transmission phase would also make the evolution more robust, like those applied in RAFT (Dong et al. 2023) and ReST (Gulcehre et al. 2023). □

B.3 Proof: LLM as a Bayesian Agent

Proposition 2. *Consider that agent A is conducting in-context learning. If the prompt examples in $\mathbf{d}^{t-1}$ are generated by another agent B with the same prior knowledge (e.g., they come from the same checkpoint and use the same prompt), sampling from the posterior predictive distribution of agent A, i.e., $d^t \sim P_{lmw}(d \mid \mathbf{d}^{t-1})$, can be decomposed into: 1.) $h^{t*} \to \operatorname{argmax}_h P_{lmw}(h \mid \mathbf{d}^{t-1})$, and 2.) $d^t \sim P_{lmw}(d \mid h^{t*})$, where h is a hidden variable that describes the mapping between x and y.*

Proof. The proof of this proposition can be divided into two steps. In the first part, we de-marginalize the posterior predictive distribution on a hidden variable h, and then show that the model automatically "selects" a hypothesis h^{t*} that generates the prompting examples. In the second part, we show that when the examples in $\mathbf{d}^{t-1}$ are generated by another LLM with the same prior belief over h, the MAP (maximize a posterior) estimation of h can approximate h^{t*} well.

Step 1: In our paper, we assume the query and answer sequences in each example, i.e., $d_i = (x_i, y_i)$, are controlled by the hidden hypothesis h, which plays a similar role to the "concept" parameter θ mentioned in Xie et al. (2022)[8]. Then the posterior predictive distribution can be decomposed as:

$$\begin{aligned} P(d \mid \mathbf{d}^{t-1}) &= \int_h p(d \mid \mathbf{d}^{t-1}, h) P(h \mid \mathbf{d}^{t-1}) \, \mathrm{d}h \\ &= \int_h p(d \mid h) P(h \mid \mathbf{d}^{t-1}) \, \mathrm{d}h \\ &\propto \int_h p(d \mid h) p(\mathbf{d}^{t-1} \mid h) P_0(h) \, \mathrm{d}h \\ &\propto \int_h p(d \mid h) \frac{p(\mathbf{d}^{t-1} \mid h)}{p(\mathbf{d}^{t-1} \mid h^{t*})} P_0(h) \, \mathrm{d}h \\ &= \int_h p(d \mid h) \exp(n \cdot r_n(h)) P_0(h) \, \mathrm{d}h, \end{aligned} \tag{13}$$

where $r_n(h) \triangleq \frac{1}{n} \log \frac{p(\mathbf{d}^{t-1} \mid h)}{p(\mathbf{d}^{t-1} \mid h^{t*})}$. In Equation (13), the second line follows the Markov property of hypothesis and data samples, the third line follows the Bayesian rule and drops a constant term, the

[8] As we only need the "selection" mechanism mentioned in this paper, the HMM (Hidden Markov Model) assumption is not required in our setting.

https://doi.org/10.52202/079017-1220

fourth line is generated by dividing a constant $p(\mathbf{d}^{t-1} \mid h^{t*})$, where h^{t*} is a hypothesis that generates $\mathbf{d}^{t-1}$. Now we see the $r_n(h)$ has almost the same form as $r_n(\theta)$ in Xie et al. (2022). By reusing the derivation in that paper (mainly in Section 3.2 and the proof of its Theorem 1), we can conclude that $\exp(n \cdot r_n(h)) \to 0$ for any $h \neq h^{t*}$ and $\exp(n \cdot r_n(h^{t*})) \to 1$. Hence Equation (13) becomes:

$$P(d \mid \mathbf{d}^{t-1}) = P(d \mid h^{t*}). \tag{14}$$

Step 2: In a general in-context learning setting, the prompting examples $\mathbf{d}^{t-1}$ are usually sampled from an unknown distribution P_{prompt}. For example, by analyzing different x_{test}, the researchers can manually design effective prompting examples sharing similar chain-of-thought structures with the target question. Obviously, such P_{prompt} is impossible to parameterize or analyze accurately. To approximate it, Xie et al. (2022) first assumes that both the prompting examples and the pre-training corpus are natural languages, and a well-trained LLM can approximate this "natural language distribution" well. Then, P_{prompt} can be well approximated by $P(\cdot \mid \theta^*)$ under some θ^*, which is the prompt concept in that paper.

In our settings, as we assume the prompting examples $\mathbf{d}^{t-1}$ are generated by another agent-B sharing the *same prior*, then the h^{t*} triggered by feeding $\mathbf{d}^{t-1}$ to agent-A would be exactly the same as that feeding that to agent-B, i.e.,

$$h^{t*} = \underset{h}{\operatorname{argmax}}\, P_{lmw}(h \mid \mathbf{d}^{t-1}), \tag{15}$$

where P_{lmw} is model's belief after receiving the common instruction $\mathbf{w}$.

Combining these two steps, we can decompose the sampling procedure $d^t \sim P(d \mid \mathbf{d}^{t-1})$ into two parts: first, inherently select h^{t*} based on observations generated by another agent in the previous generation; then sample new d conditioned on this h^{t*}, which matches the Bayesian-IL procedure discussed in this paper. □

C Experiments of Iterated Learning on Different Domains

To provide a panorama of iterated learning and its applications in different fields, this appendix will first give an intuitive explanation using some lab experimental results. Then, experiments on the emergence of compositional language among Bayesian agents are introduced to verify all the theoretical hypotheses.

C.1 Iterated Learning in Lab Experiments using Lewis Language Game

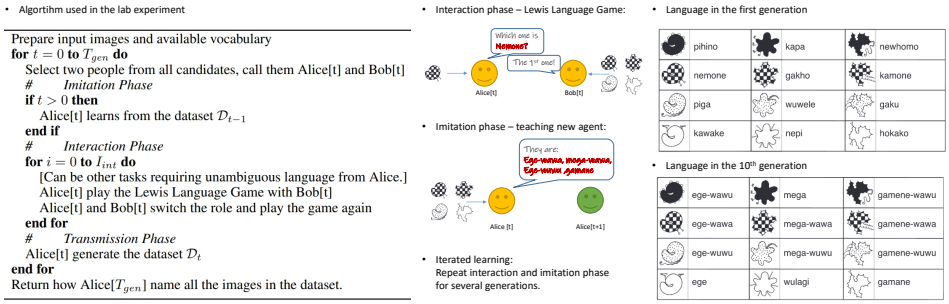

Figure 6: The lab experiments (algorithm, settings, and results) conducted in Kirby et al. 2015.

As denoted in Kirby et al. 2008, iterated learning is a process where one individual learns by observing the output of another individual, who learned in the same way. In this multiple-generation learning procedure, the shared language (i.e., h) among learning agents will gradually become more systematic under the compressibility pressure (imposed during imitation, embodied in $P_0(h)$) and expressivity pressure (imposed during interaction, requiring $h \in \mathcal{H}_{\text{eff}}$). To simulate this process, authors of Kirby et al. 2015 design a two-phases learning procedure illustrated in Figure 6. In the interaction phase,

the speaker (Alice) and listener (Bob) must cooperate to accomplish a Leiws referential game (see the middle panel in Figure 6). Specifically, Alice will first create a name for the given objects and talk that to Bob. After receiving this message, Bob needs to select the correct object shown to Alice among some candidates. If correct, both of them are rewarded. This phase terminates when they can achieve a high enough success rate. To succeed in this game, the shared language should be expressive enough to avoid any ambiguities – we expect the language to be a bijection. Then, we select another new naive candidate (Alice[t+1]) and let it learn the naming system created by Alice[t] and Bob[t]. In this phase, those highly structural mappings should be easier for a human to remember, which is how the compressibility pressure is imposed. After that, Alice[t+1] will play the same game with another Bob[t+1] and the interaction phase starts again. We provide the languages generated by Alice[0] and Alice[10] in the right panel in Figure 6: it is clear that an interesting structure emerges in the language generated by Alice[10]. There are also plenty of similar lab experiments that support the "two pressures" and cultural evolution hypothesis using IL-like training methods, e.g., Fay et al. 2010; Ferdinand et al. 2019; Motamedi et al. 2019; Motamedi et al. 2022. Although these methods have different types of input, game designs, learning procedures, vocabularies (like gesture language), etc., the conclusion of them is quite consistent: compressibility and expressivity pressures are crucial for the emergence of systematic mappings, iteratively learning and interacting can amplify these pressures a lot, which matches the Bayesian explanations well.

C.2 Iterated Learning of Bayesian Agents (re-implementation of results in Kirby et al. (2015))

To verify that the emergence of systematic mappings in iterated learning is not an accident, authors of Beppu and T. Griffiths 2009 provide a guarantee by analyzing the behavior of Bayesian agents. There are also plenty of related works in cognitive science, like Reali and T. L. Griffiths 2009; Perfors et al. 2011; Bonawitz et al. 2014; Ferdinand et al. 2014; Navarro et al. 2018, discussing the influence of and theories behind iterated learning and Bayesian analysis.

To give the readers a better understanding of how iterated learning works, we re-implement the Bayesian experiments mentioned in Kirby et al. 2015. Consider the following toy example, where we have four different input objects: $\mathcal{X} = \{\texttt{blue circle, blue box, red circle, red box}\}$, and four possible names: $\mathcal{Y} = \{\texttt{00,01,10,11}\}$. The hypothesis h is defined as $h \in \mathcal{H} : \mathcal{X} \to \mathcal{Y}$. In this example, we have $|\mathcal{H}| = 256$, which means $P(h)$ can be parameterized by a categorical distribution with 256 dimensions. In this analysis, we assume the prior distribution of a mapping is negatively correlated with its *coding length* α, i.e., $P(h; \alpha, c) \propto 2^{-\frac{\alpha}{c}}$, where c is a normalizing constant to make sure the prior distribution is not too peaky. Usually, the easier-to-learn mappings (i.e., more systematical ones) have higher prior. In Table 3, we demonstrate how to calculate the coding length for the three typical mappings. Note that the mapping that has the highest prior is a degenerate mapping, where $\alpha = 18$ and $P(h) \approx 0.6$. The $P_0(h)$ for all possible mappings are demonstrated in Figure 7.

Table 3: An example of coding the mappings, where α is how many characters (including space and unique symbol, e.g., $\to$ and :) are used to express the grammar.

A systematic mapping $\alpha = 43$				A holistic mapping $\alpha = 56$				A degenerate mapping $\alpha = 18$		
S		$\to$	z2, z1							
z2:	0	$\to$	blue	S:	00 $\to$	blue circle				
z2:	1	$\to$	red	S:	01 $\to$	red circle		S:	00 $\to$	Everything
z1:	0	$\to$	circle	S:	10 $\to$	red box				
z1:	1	$\to$	box	S:	11 $\to$	blue box				

In this experiment, the knowledge of an agent is encoded in its posterior distribution, i.e., $P_{lm}(h)$. We will observe how this distribution evolves when the agents conduct iterated learning. Recall the sampling behaviors discussed in Section 3. To get a data sample $d = (x, y)$, we first randomly sample $x \in \mathcal{X}$ from a uniform distribution and then sample y based on the given x,

$$d \sim P_{lm}(x, y) \propto P_{lm}(y \mid x) \propto p(y \mid h, x) \cdot P_{lm}(h). \tag{16}$$

https://doi.org/10.52202/079017-1220

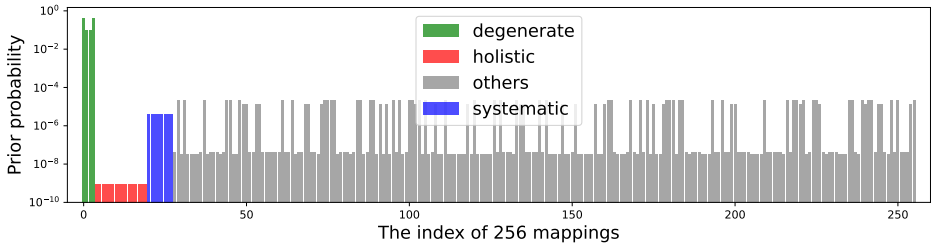

Figure 7: The prior probability of all possible $h \in \mathcal{H}$. The systematic mappings are sandwiched between degenerate and holistic mappings, which means $P_0(h_{degen}) > P_0(h_{sys}) > P_0(h_{holi})$. Some mappings in the "other" group also have relatively large prior, because they contain degenerate components (e.g., mapping two or three objects to the same message).

The likelihood $p(y \mid h, x)$ is defined as:

$$p(y \mid h, x) = \begin{cases} 1 - \epsilon & \text{if } y \text{ is mapped to } x \text{ in } h \\ \dfrac{\epsilon}{|\mathcal{Y}| - 1} & \text{otherwise,} \end{cases} \tag{17}$$

where ϵ is a small positive value describing the systematic error during communication.

For the learning behavior, the agent will update the posterior based on the received data samples $\mathbf{d} = (x_i, y_i)_{i=1}^N$:

$$P_{lm}(h) = P(h \mid \mathbf{d}) \propto p(\mathbf{d} \mid h) \cdot P_0(h) \propto P_0(h) \cdot \prod_{i=1}^{N} p(y_i \mid h, x_i). \tag{18}$$

Now, with the definition of learning and sampling for these Bayesian agents, we can describe how they conduct IL:

- Initialization: at the beginning of the t-th generation, a new agent is initialized by $P_0(h)$
- Imitation: agent-t learns from $\mathbf{d}^{t-1}$, which is generated by agent in the previous generation following Equation (18)
- Interaction: to impose expressivity pressure, we let agent-t (Alice) play a communication game in this phase. Specifically, we first create another agent Bob by copying $P_{lm}(h)$ from Alice. Then, Alice samples a data pair $d = (x, y)$ on a randomly chosen x and sends it to Bob. Bob will estimate the object based on y. If the estimated $x' = x$, the game succeeds and data pair (x, y) is added to a buffer named $\mathbf{d}_{\text{comm}}$. After several rounds, Alice updates its knowledge by learning from $\mathbf{d}_{\text{comm}}$. Note that in this phase, the pressure of $h \in \mathcal{H}_{\text{eff}}$ is induced implicitly: for the ambiguous h, where multiple x are mapped to the same y, Bob's reconstruction x' might not equal x with high probability. Hence $\mathbf{d}_{\text{comm}}$ will finally dominated by the samples generated by those $h \in \mathcal{H}_{\text{eff}}$.
- Transmission: after the interaction phase, Alice will generate multiple samples $\mathbf{d}^t$ for the next generation.

In the above procedure, the compressibility pressure is embodied in the prior distribution where more reusable principles lead to a higher probability, which aligns with the simplicity bias in the human cognition system. The expressivity pressure is imposed in the communication game because $\mathbf{d}_{\text{comm}}$ only contains the unambiguous mappings. Under this setting, we can calculate the weighted proportion (i.e., the summation of the posteriors) of different types of languages and observe how they evolve during iterated learning, as illustrated in Figure 8. It is clear that the systematic mappings gradually dominate as the learning progresses.

To further verify our theory, we consider a compressibility-only case by removing the interaction phase, and an expressivity-only case using a uniform prior $P(l) = 1/256$. The results in Figure 8 match the theory quite well:

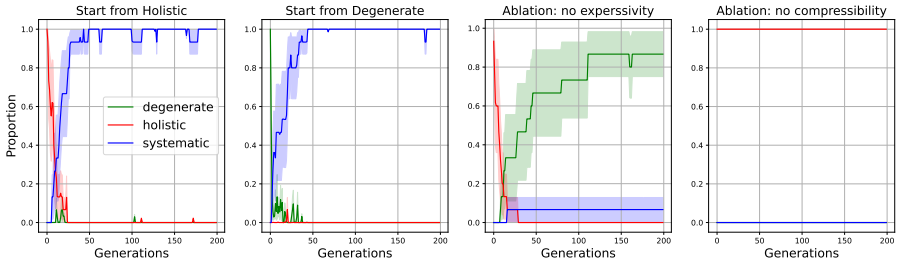

Figure 8: Ratio of three different types of mappings during iterated learning (curves are the average of 15 different runs, shadow region is the variance). Left to right: 1.) $\mathbf{d}^0$ is a holistic mapping; 2.) $\mathbf{d}^0$ is a degenerate mapping; 3.) Starting from a holistic $\mathbf{d}^0$, but no longer conduct an interaction phase during training. Hence the degenerate language, which has the highest prior, will gradually dominate; 4.) Ablating the compressibility pressure by using a uniform prior distribution.

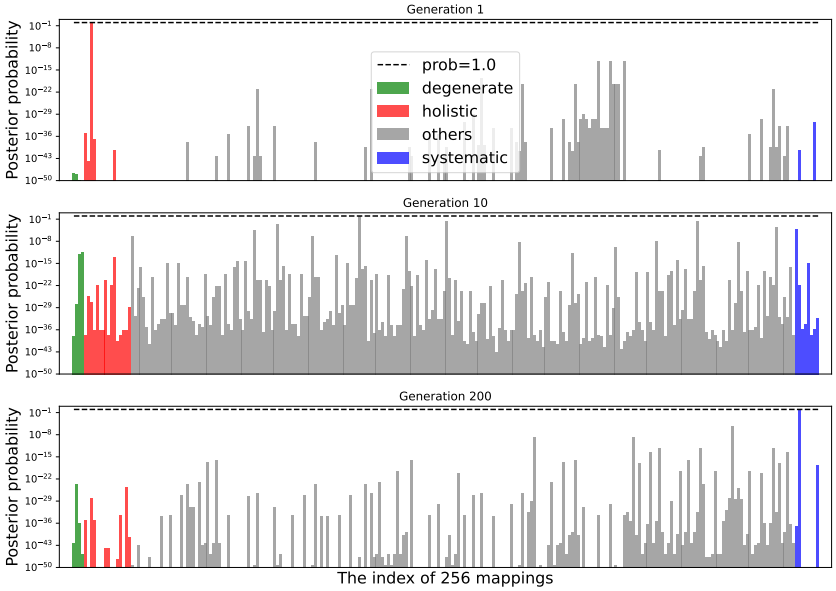

Figure 9: The posterior probabilities of all at the end of different generations.

- In an imitation-only iterated learning case, i.e., the third panel in Figure 8, h^T converges to a degenerate mapping, which has the highest prior as illustrated in Figure 7;

- Introducing the interaction phase will rule out those ambiguous mappings (i.e., those $h \notin \mathcal{H}_{\text{eff}}$), and hence h^T converges to systematic mapping, which has the highest prior among all $h \in \mathcal{H}_{\text{eff}}$;

- By comparing the first and second panels in Figure 8, h^T always converges to systematic mapping no matter $\mathbf{d}^0$ is holistic or degenerate.

Furthermore, we can directly observe the dynamics of $P_{lm}(h)$ from Figure 9, which provides a more detailed illustration of how the posterior of all mappings changes during training. In the first generation, we see the dominant mapping is a holistic one, which is our $\mathbf{d}^0$. Then gradually, under the two pressures, the posterior of systematic mappings gradually increases and finally dominates.

https://doi.org/10.52202/079017-1220

D More on GPT-based ACRE Experiments

D.1 How to Calculate the Model's Posterior on All Hypotheses (Figure 10)

Thanks to the instruction-following ability, the GPT can always provide responses following the given format, as illustrated in Figure 10. The experiments demonstrated in this part come from OpenAI's playground. The model we use is `gpt-3.5-turbo-instruct`. The temperature is 0.1 and the probability feedback is enabled. We let the model return probabilities of the top 5 candidate tokens for each token in the response, as illustrated by the three sub-panels in the figure. Then, the posterior of specific h can be calculated by multiplying the probability of all tokens with corresponding values. For example, $P_{lmw}(h = \texttt{\{A:on, B:und, C:off, D:...\}})$ can be calculated by $P(r_5 = \texttt{on}) \cdot P(r_9 = \texttt{und}) \cdot P(r_{13} = \texttt{off}) \cdots$, where $r_{5,9,13,...}$ are the tokens denoting the corresponding values of `A, B, C`. To further show the feasibility of this approach, we conduct the following two verifications. First, we calculate $\prod_{i \in \mathcal{I}_{\text{format}}} P(r_i)$ and find that this value is always close to one ($\mathcal{I}_{\text{format}}$ denotes the indexes of those format-related tokens, e.g., `Rule`, `{`, `:`, `A`, `B`, `C`, etc). This means the model reliably follows the instructions when generating responses. Second, we calculate $P_{lmw}(h)$ for all possible 243 different h and verified that $\sum_{h \in \mathcal{H}} P_{lmw}(h)$ is always close to one.

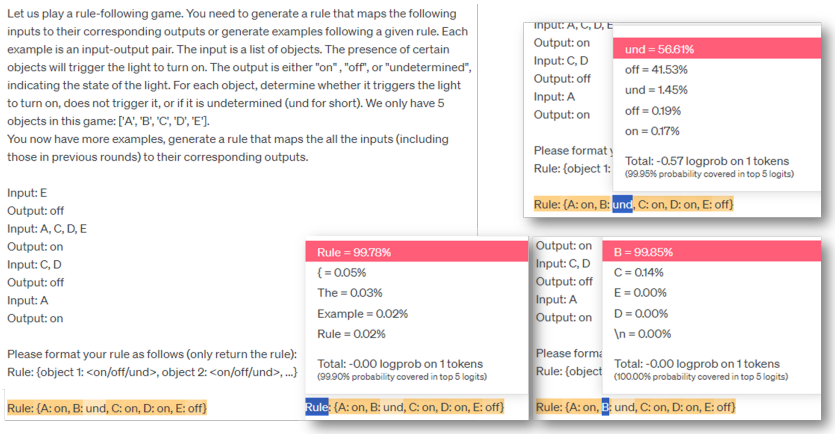

Figure 10: How GPT provides the rule following the given format, which makes it possible to calculate $P_{lmw}(h)$ for all h.

D.2 How to Control the Bias in the Prior

We demonstrate how to manipulate bias in the prior $P_0(h)$ in the ACRE task by adding spurious correlations in the prompt and by changing the name of the object. The prompt we use is almost the same as that in Figure 10. We will analyze the value of $P(r_{20})$, i.e., the probability of the token denoting the status of the last object in the rule. The $\mathbf{d}^0$ in these experiments are all the same (i.e., `[B,D]` $\rightarrow$ `on`; `[B,C]` $\rightarrow$ `und`; `[B,D,E]` $\rightarrow$ `on`, as stated in the last several panels in Figure 11). Some subtle modifications under different settings will be described one by one in the following:

⓪: the default setting, where the object list is `[A,B,C,D,E]`. The $P_0(r_{20} = \texttt{[on,off,und]})$ are $[11.8\%, 6.46\%, 81.68\%]$. This makes sense, as all these 3 statuses of `E` can describe all examples in $\mathbf{d}^0$

①: compared with ⓪, we change the object `E` to `screen`, and no extra text is added to the prompt. Then, as the screen is likely to be turned on during the experiment, $P_0(r_{20} = \texttt{on})$ dominates the prediction;

②: compared with ①, we add a sentence "Turn off the screen after the experiment" to the task instruction. This misleading sentence introduces a bias towards `screen:off` by creating a spurious correlation;

③: compared with ②, we use a synonym "close the screen" to replace the "turn off the screen" in the prompt. As the word "off" does not exist in the prompt, the bias towards `screen:off` is weakened;

④: compared with ②, we change the name `screen` to `Sony screen` in the example, but left the prompt unchanged. We see the model is clever enough to distinguish which screen we refer to, and hence keeps the preference of $P_0(r_{20})$ demonstrated in ①;

⑤: here we change the object to another name `John`, which also keeps the preference of $P_0(r_{20})$ demonstrated in ①;

⑥: compared with ⑤, we add the sentence "John will turn off the screen after experiment". Then we find the bias towards `John:off` is slightly increased, but not as strong as that in ②, which provides us another way to control the strength of the bias;

⑦: in the following three cases, we put the position of the misleading sentence before the examples $\mathbf{d}^0$. Compared with ②, the bias towards `screen:off` is significantly amplified. This might be because the attention mechanism lets the model recite the fact that the screen is off before reading the examples (remember that `screen:off` also explains all examples);

⑧: compared with ④, where the object is also `Sony screen`. Here the bias is stronger than ④ but weaker than ⑦, which verifies that adding spurious correlation before examples can amplify the bias while modifying the object name can reduce the bias;

⑨: compared with ③, which also uses the synonym in the prompt, the bias becomes stronger.

In summary, we have several principles when controlling the strength of the bias in $P_0(h)$:

- Adding spurious correlation before $\mathbf{d}^0$ provide very strong bias;
- Using synonyms rather than phrases containing specific states (e.g., close/open v.s. turn off/on) weakens the bias;
- Using two slightly different object names (i.e., Sony screen v.s. screen) weakens the bias;
- Using indirect spurious correlation (e.g., John v.s. screen) weakens the bias.

Figure 11: An example of how to manipulate $P_0(h)$ by adding spurious correlation in the prompt. The task instruction is the same as that provided in Figure 10, except the object name (in the blue box) and the added hints (in the red box).

For the experiments in Figure 13 and Figure 3, the six different levels of prior bias are controlled by the following prompts:

- Very high: add "`Turn off the screen after the experiment.`" before and after $\mathbf{d}^t$ is given;
- High: add "`Turn off the screen after the experiment.`" before $\mathbf{d}^t$;
- Medium: add "`Turn off the screen of the monitor after the experiment.`" before $\mathbf{d}^t$;
- Mild: add "`John will turn off the screen after the experiment.`" before $\mathbf{d}^t$;
- Low: add "`Close the screen after the experiment.`" before $\mathbf{d}^t$;
- Very low: add "`Close the screen after the experiment.`" after $\mathbf{d}^t$;

https://doi.org/10.52202/079017-1220

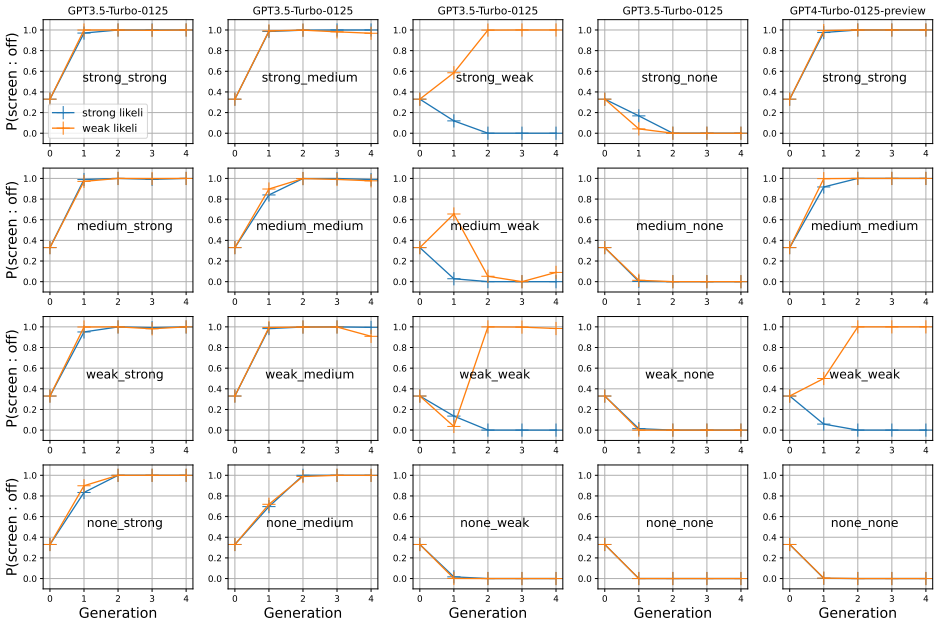

Figure 12: A more detailed analysis of the competition between the likelihood $p(\mathbf{d} \mid h)$ and bias in $P_0(h)$. The first 4 columns are results from the new `GPT3.5` and the last column is from `GPT4`. In each panel, the two curves represent strong and weak likelihood cases, which are controlled by the ground truth h^*. The h^* in strong cases contains 3 objects being `"on"` while the weak cases only have 1. The text in each panel represents the level of spurious correlation we introduce before and after $\mathbf{d}$ by manipulating the instruction prompt. For example, `strong_strong` means we put strong bias, i.e., `"Turn off the screen after experiments"`, before and after $\mathbf{d}^t$ in each generation. The trend of these panels aligns well with our previous results and analysis.

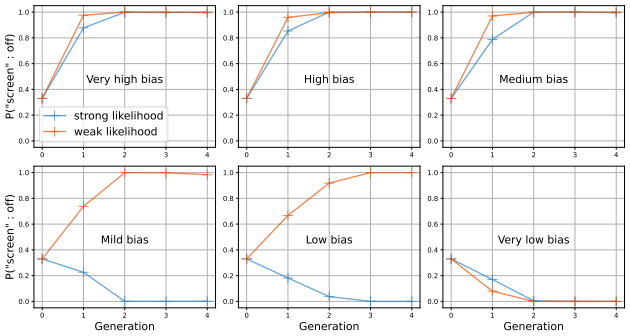

Figure 13: The follow-up experiments of the one mentioned in Figure 3. In this figure, we put curves of the same level of prior bias (but with different likelihoods) in the same panel. It is clear that in most cases, the stronger likelihood will weaken the influence of the bias (that is why the orange curve is above the blue curve).

D.3 The Prompt Design for ACRE Task (Figures 14 to 16)

Please also refer to the three figures and the `log.txt` file in our code base.

E More on Self-data-augmentation Task (Table 4 and 5; Figure 17, 18, and 19)

In this appendix, we first introduce the prompt design of the experiment on both imitation-only settings, as well as the experiments with five different $\mathcal{H}_{\text{eff}}$. Please refer to Figure 17 for more details.

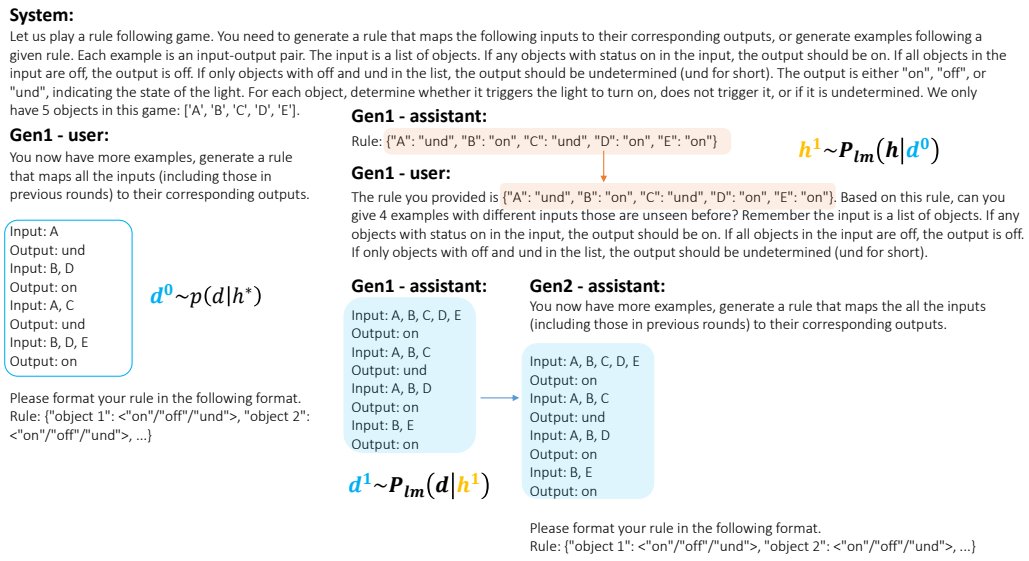

Figure 14: Prompt design and an example dialogue for the imitation-only iterated learning on the ACRE dataset. The shaded region and arrows represent that we copy specific text to form the message. The messages starting with the role of **system** and **user** are sent to GPT, while those starting with **assistant** are the feedback from GPT. For this multi-round chat, we will feed all historical information to API, see our code for more details.

Then, we provide the full results of the experiment in Table 4, 5 and Figure 18, from which we can derive more interesting findings. In general, the figure demonstrates how different metrics evolve for different generations while the table reports the converged values at the last generation.

From the first column in this figure, we observe that other than the $\mathcal{H}_{\text{hard}}$ setting, all curves in other settings show a clear trend of convergence towards the top of the figures. This means iterative learning indeed amplifies the hidden bias of $P_0(h_{\text{easy}}) > P_0(h_{\text{hard}})$ when the imposed $\mathcal{H}_{\text{eff}}$ doesn't impede this bias. In the $\mathcal{H}_{\text{random}}$ setting, the $\hat{\mathbf{d}}$ is sampled from all the data generated by the previous generation, which makes those hard samples more likely to be sampled compared with the imitation-only settings or those with $\mathcal{H}_{\text{easylong/short}}$. Hence we observe the converging speed of it is slightly lower than these settings. On the contrary, when $\mathcal{H}_{\text{hard}}$ is introduced, the bias towards h_{easy} is successfully restrained. We observe a clear competition between these two pressures: the curve first goes up, which means the bias towards easy samples is stronger. However, as the learning goes on, the curves turn down again as we later have more hard samples in $\mathcal{D}_{\text{pool}}$.

The second column of the figure demonstrates the average ranking of words in $\mathbf{d}^t$. We observe a similar trend in the ratio of easy samples, although our $\mathcal{H}_{\text{eff}}$ never explicitly constrains it. **This phenomenon hints to us that when conducting an iterative self-data-augmentation algorithm, some unknown bias would be implicitly amplified although we already designed another $\mathcal{H}_{\text{eff}}$ for other properties.** Imagine we are conducting the ReST algorithm (Gulcehre et al. 2023). We can pursue the correctness of $\mathbf{d}^t$ by ranking all examples by training a reward model that prefers more correct responses. However, some other subtle biases, like conciseness, informativeness, etc., might be ignored by the algorithm designer and are hence unexpectedly amplified. In summary, we should bear in mind that identifying the good and bad bias in $P_0(h)$ is quite important for an appropriate evolution.

Finally, we use the last column and the average length of the acronym to show how to make a composed $\mathcal{H}_{\text{eff}}$ by combining more than one attribute of the data. It is clear that both $\mathcal{H}_{\text{easylong}}$ and $\mathcal{H}_{\text{easyshort}}$ did their jobs quite well: the converged $\mathbf{d}^6$ contains the samples with desired properties as we expected. Another thing that heavily influences the results is the ratio of easy examples in $\mathbf{d}^0$. Although the theory claims that the converged results are irrespective of $\mathbf{d}^0$, the converging speed and the difficulty of amplifying specific bias heavily depends on $\mathbf{d}^0$. This claim can be well supported by

https://doi.org/10.52202/079017-1220

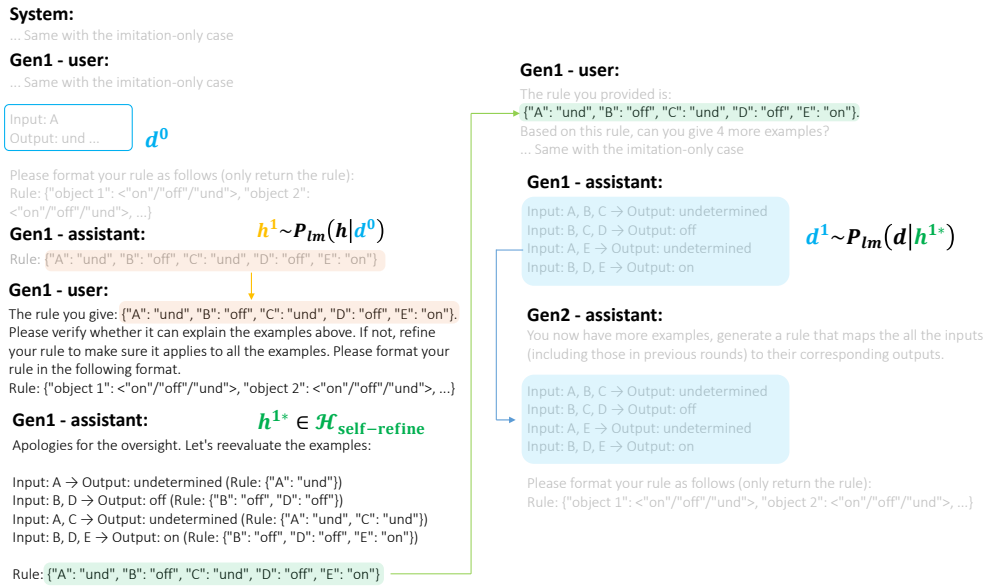

Figure 15: Prompt design for iterated learning with self-refine (Madaan et al. 2023) as the interaction phase. The text in gray is the same as the default imitation-only setting. Note that the format of the examples might be changed (like the examples in Gen-1: assistant), which doesn't influence the experimental results.

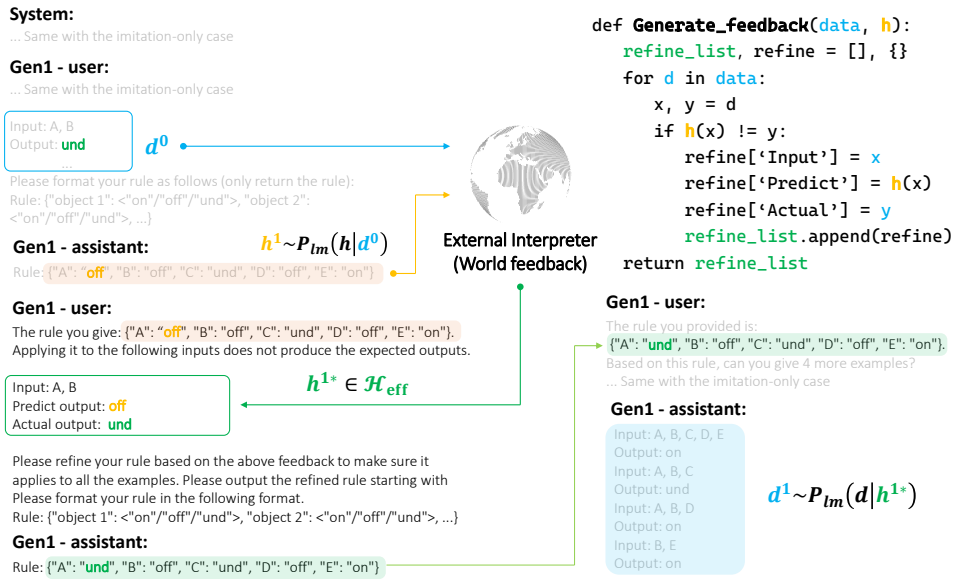

Figure 16: Prompt design for iterated learning with hypothesis search (Qiu et al. 2024) as the interaction phase. Compared with self-refine, it introduces an external interpreter to refine h proposed by the model, where $\mathcal{H}_{\text{eff}}$ is the ground-truth one.

the fact that when N_e is small, amplifying the bias of h_{easy} is significantly harder than the large N_e case.

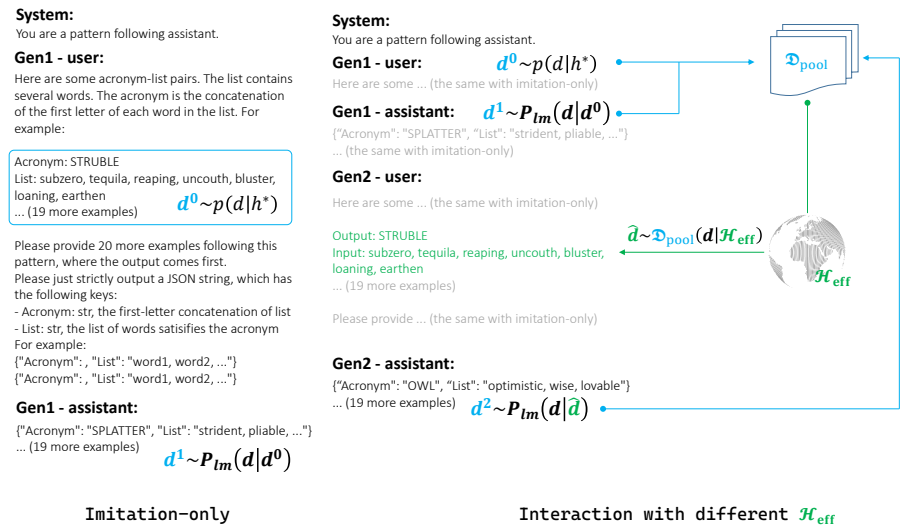

Figure 17: Prompt design for iterated learning on the acronym data-generation task.

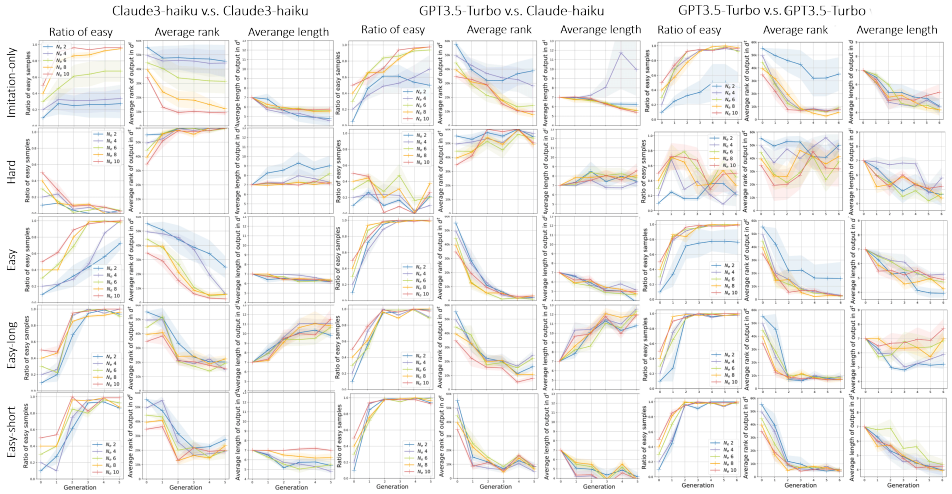

Figure 18: Results when adding different interaction phases (4 different seeds). All three settings demonstrate similar evolutionary trends, which match our theory quite well.

Figure 19: An interesting observation of Mixtral series models: they have a bias toward alphabet examples. However, as the Mixtral model usually has typos in their response (like the right panel), we do not have the full results of these models.

https://doi.org/10.52202/079017-1220

Table 4: Claude3-haiku results under different $\mathcal{H}_{\text{eff}}$. We color the highest and lowest numbers in each column differently.

	Ratio-easy					Avg-rank					Avg-length				
Ne=	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
Imitation-only	0.275±0.18	0.338±0.17	0.675±0.13	0.950±0.00	0.962±0.00	45477	44235	31356	12148	9272	**4.788±2.21**	**4.563±0.14**	5.613±0.46	**5.563±0.15**	**5.775±0.05**
Hard	**0.036±0.00**	**0.000±0.00**	**0.038±0.00**	**0.000±0.00**	**0.025±0.00**	**59484**	**60001**	**59458**	**60001**	**59874**	9.014±4.32	7.288±0.20	8.175±2.89	7.238±0.11	7.137±0.12
Easy	0.725±0.08	**1.000±0.00**	**0.975±0.00**	**0.988±0.00**	**1.000±0.00**	24809	**6233**	**5644**	**5827**	**2646**	6.188±0.25	6.300±0.04	6.338±0.09	6.375±0.07	6.225±0.22
Easylong	**0.938±0.00**	**1.000±0.00**	0.913±0.02	0.963±0.00	**1.000±0.00**	**20208**	14803	19200	22333	15708	**9.825±0.37**	**11.077±0.82**	**10.650±2.47**	**10.813±1.41**	**11.513±3.97**
Easyshort	0.863±0.00	0.925±0.00	0.925±0.02	0.875±0.00	0.988±0.00	27286	19021	19749	23155	19979	5.413±0.60	4.800±0.14	**5.475±0.23**	6.225±0.68	6.988±0.01

Table 5: GPT3.5-Turbo 0125 plays with Claude3-haiku results when adding different $\mathcal{H}_{\text{eff}}$.

	Ratio-easy					Avg-rank					Avg-length				
Ne=	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
Imitation-only	0.363±0.11	0.575±0.14	0.975±0.00	**1.000±0.00**	0.975±0.00	44334	33977	10960	6398	9688	6.363±0.39	7.705±5.25	4.475±0.26	5.134±0.20	5.012±0.22
Hard	**0.000±0.00**	**0.025±0.00**	**0.110±0.01**	**0.000±0.00**	**0.000±0.00**	**60001**	**59382**	**57022**	**60001**	**60001**	8.225±2.76	6.700±0.27	8.606±0.90	7.950±0.69	8.063±1.18
Easy	**1.000±0.00**	**1.000±0.00**	**1.000±0.00**	**1.000±0.00**	**1.000±0.00**	**777**	**1552**	**1341**	**1048**	**1850**	4.750±0.11	5.350±0.53	5.175±0.19	4.888±0.07	4.975±0.32
Easylong	0.988±0.00	**1.000±0.00**	**1.000±0.00**	**1.000±0.00**	**1.000±0.00**	15967	10009	9087	7432	6146	**11.025±0.54**	**10.86±0.52**	**10.525±1.99**	**11.975±2.03**	**10.875±0.59**
Easyshort	0.975±0.00	**1.000±0.00**	**1.000±0.00**	0.913±0.02	**1.000±0.00**	14139	7848	15015	16249	12356	**4.163±0.90**	**3.213±0.12**	**3.513±0.22**	**3.825±0.09**	**3.887±0.45**

https://doi.org/10.52202/079017-1220

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Claims are justified.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss this throughout.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

https://doi.org/10.52202/079017-1220

Justification: Theorems include full statements and proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Code to reproduce the experiments is provided.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code is publicly released.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We describe settings in general in the main body, and details are provided in the appendix and in the published code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Results have error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

https://doi.org/10.52202/079017-1220

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: We did not carefully track our computational resources, but they were mostly limited to modest numbers of API calls to existing LLMs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We follow the code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The paper is about the evolution of LLM agents and potentially steering that evolution; this has roughly the same potential for impact as any LLM work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not release any novel datasets or models.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Datasets, models, etc. are cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

https://doi.org/10.52202/079017-1220

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code for experiments is fairly simple, with documentation inline.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.