ETHER: Aligning Emergent Communication for Hindsight Experience Replay

Kevin Denamganaï, Daniel Hernandez, Ozan Vardal, Sondess Missaoui, and James Alfred Walker
Department of Computer Science
University of York
York, UK
kevin.denamganai@york.ac.uk,
daniel.hernandez@sony.com, ozan.vardal@york.ac.uk,
sondess.missaoui@york.ac.uk, james.walker@york.ac.uk

Abstract

Natural language instruction following is paramount to enable collaboration between artificial agents and human beings. Natural language-conditioning of reinforcement learning (RL) agents has shown how natural languages' properties, such as compositionality, can provide a strong inductive bias to learn complex policies. Previous architectures like HIGhER combine the benefit of language-conditioning with Hindsight Experience Replay (HER) to deal with sparse rewards environments. Yet, like HER, HIGhER relies on an oracle predicate function to provide a feedback signal highlighting which linguistic description is valid for which state. This reliance on an oracle that must be provided by the user or benchmark limits its application. Additionally, HIGhER only leverages the linguistic information contained in successful RL trajectories, thus hurting its final performance and data-efficiency. Without early successful trajectories, HIGhER is no better than DQN upon which it is built.

In this paper, we propose the Emergent Textual Hindsight Experience Replay (ETHER) agent, which builds on HIGhER and addresses both of its limitations by means of (i) a discriminative visual referential game, commonly studied in the subfield of Emergent Communication, used here as an unsupservised auxiliary task and (ii) a semantic grounding scheme to align the emergent language with the natural language of the instruction-following benchmark. We show that the speaker and listener agents of the referential game make an artificial language emerge that is aligned with the natural-like language used to describe goals in the BabyAI benchmark and that it is expressive enough so as to also describe unsuccessful RL trajectories and thus provide feedback to the RL agent to leverage the linguistic, structured information contained in all trajectories. Our work shows that emergent communication is a viable unsupervised auxiliary task for goal-conditioned RL in sparse reward settings and provides missing pieces to make HER more widely applicable.

1 Introduction

Since time immemorial, natural languages have been harnessed by humans as powerful tools to describe not only reality as one senses it, but also as one imagines it (e.g. via the poetic function of languages [31]). Through properties such as compositionality and recursive syntax natural languages become flexible interfaces that allow humans to express arbitrarily complex meanings. Beyond being immensely useful for inter-human communication, natural languages can also be a fruitful means of communication between humans and AI models, as recently showed by the advent of large language models [63]. The use of natural languages to condition the behaviour of reinforcement learning (RL) agents remains an open question with an untapped potential [45]. In its most general form, how to train RL agents capable of achieving an arbitrary set of goals is the fundamental question within goal-conditioned RL. Language-conditioned RL addresses the challenge of training agents to attain a broad array of objectives, utilizing natural languages as an expressive and intuitive tool to define those goals.

To be able to describe goals in natural language and have language-conditioned RL agents learn well performing policies is already very useful. However, even optimal policies might not always be able to accomplish a task. For instance, a cleaning robot might not being able to wash the dishes in the sink if there is no soap left. In this scenario, to close the human-AI communication loop an agent could communicate that it succeeded at other goals such as *pick up a plate* and *turn water tap on*. This capability would greatly contribute towards agent explainability, yet it posits a hard question to answer: how may the agent learn in an unsupervised manner a communication protocol whose semantics align with the semantics of the language used to describe the goals it trains on? In other words, how may an agent learn to communicate whether it succeeded at *pick a plate* when it initially has no notion of what a *plate* is and what it means to *pick*?

To tackle the challenge of language-conditioned RL and the learning of aligned emergent communication protocols we present Emerging Textual Hindsight Experience Replay (ETHER). Agents trained with ETHER learn a function mapping observed states to goals that the agents have reached, a missing piece in current language-conditioned RL, and further improves on the sample efficiency its state-of-the-art counterparts.

Our main contributions are threefold. Firstly, we extend HIGhER by enabling its deployment in any instruction-following task out-of-the-box without relying on any oracle. This is achieved by means of a learned, approximate predicate function, which we detail in Section 3.1. Secondly, in order to further leverage unsuccessful trajectories, we propose the Emergent Textual Hindsight Experience Replay (ETHER) architecture, which builds on HIGhER and addresses both of its limitations, showing that a discriminative visual referential game is a viable unsupervised auxiliary task for RL [30]. This is detailed in Section 3.2. Finally, facing the common problem of the emergent language shifting from natural languages, despite the instruction-following task making use of a natural-like language, we show that it is possible to align to some extent the emergent language with the natural language of the instruction-following benchmark by leveraging the semantic co-occurrence of visual and textual concepts. Taken together, our work shows that emergent communication is a viable unsupervised auxiliary task for goal-conditioned RL in sparse reward settings and provides missing pieces to make HER more widely applicable.

We continue by reviewing necessary background and notation in Section 2. After delineating our methods in Section 3.1, we present experimental results on the PickUpDist instruction-following task of the BabyAI benchmark [13] in Section 4. Importantly, our results demonstrate that on a $200k$ observation budget our final agent method achieves almost twice the performance of the baseline HIGhER. Finally, we conclude in Section 5.

2 Background & Notation

2.1 Goal-Conditioned Reinforcement Learning

In goal-conditioned RL, a goal-conditioned agent makes use of a policy $\pi : \mathcal{S} \times \mathcal{G} \rightarrow \mathcal{A}$ to interact at each time step t with an environment to maximize its cumulative discounted reward over each episode $\sum_t \gamma^t r(s_t, a_t, s_{t+1}, g_t)$, where $\gamma \in [0, 1]$ is the discount factor, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{G} \rightarrow \mathbb{R}$ is the environment-defined goal-conditioned reward function over the state space $\mathcal{S}$, action space $\mathcal{A}$, and goal space $\mathcal{G}$. It interacts by choosing an action $a_t \in \mathcal{A}$ based on the state $s_t \in \mathcal{S}$ it is in and a predefined goal $g \in \mathcal{G}$ sampled at the beginning of the episode. Along with the output of the reward function, the agent is provided at each interaction with the next state s_{t+1} sampled from the transition distribution $T(s_{t+1}|s_t, a_t)$. We employ a goal-conditioned Q-function, i.e. Universal Value Function Approximator [59], defined by $Q_\pi(s, a, g) = \mathbb{E}_\pi[\sum_t \gamma^t r(s_t, a_t, s_{t+1}, g)|s_0 = s, a_0 = a, s_{t+1} \sim T]$ for all $(s, a, g) \in \mathcal{S} \times \mathcal{A} \times \mathcal{G}$. While previous works makes use of Deep Q-learning (DQN)[46] to evaluate the Q-function with neural networks and perform off-policy updates by sampling transitions $(s_t, a_t, r_t, s_{t+1}, g)$ from a replay buffer, we employ Recurrent Replay with Distributed DQN (R2D2) [33].

2.2 Hindsight Experience Replay and its Limitations

In the context of goal-conditioned RL, rewards are inherently sparse for any given goal and this is exacerbated the larger the goal space $\mathcal{G}$ is. In order to alleviate these issues, Andrychowicz et al. [2] proposed Hindsight Experience Replay (HER) which involves relabelling unsucessful (null-reward) trajectories where the agent failed to reach the sampled goal $g \in \mathcal{G}$, with a new goal $g' \in \mathcal{G}$ that is actually found to be fulfilled in the final state of the relabelled episode. This approach improves the sample efficiency of off-policy RL algorithms by taking advantage of failed trajectories, repurposing them by reassigning them to the goals that were actually achieved.

In effect, for each unsuccessful trajectory, the agent's memory/replay buffer is updated with one negative trajectory and an additional positive (relabelled) trajectory. In order to do so, HER assumes the existence of a mapping/re-labelling function $m : S \rightarrow G$, which is an oracle (i.e., externally providing expert knowledge of the environment to the algorithm). It maps a state s onto a goal g that is achieved in this state. As their experiments deal with spatial goals, vanilla HER can extract the re-labelling goal from the achieved state (because $\mathcal{G} = \mathcal{S}$), but in the more general case, it cannot be applied without an external expert as this re-labelling oracle cannot be derived. HER's need for expert interventions drastically reduces its interest and range of applicable use cases.

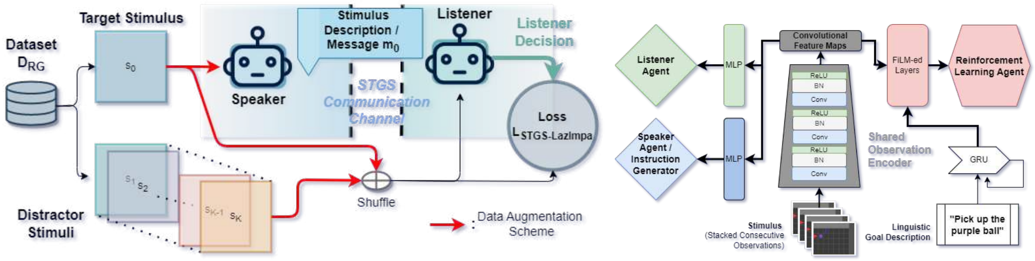
HER also assumes the existence of a predicate function $f : \mathcal{S} \times \mathcal{G} \rightarrow \{0, 1\}$ which encodes whether the agent in a state s satisfies a given goal g. This predicate function f is used to define the **learning reward function** $r_{learning}(s_t, a_t, s_{t+1}, g) = f(s_{t+1}, g)$, that is used to infer the reward at each timestep of the re-labelled trajectories. Indeed, while at the beginning of an episode a goal g is drawn from the space $\mathcal{G}$ of goals by the environment and, at each time step t, the transition $(s_t, a_t, r_t, s_{t+1}, g)$ is stored in the agent's memory/replay buffer with the rewards coming from what we will refer to as the **behavioral reward function**, i.e. the reward function instantiated by the environment, re-labelling involves using another reward function: at the end of an unsuccessful episode of length T, re-labelling and reward prediction occurs in order to store a seemingly-successful (relabelled) trajectory: an *alternative* goal $\hat{g}^0$ and corresponding reward sequence $(r^0_t)_{t \in [0,T]}$ are inferred using the learning reward function (detailed below). New transitions $(s_t, a_t, r^0_t, s_{t+1}, \hat{g}^0)_{t \in [0,T]}$ are thus added to the replay buffer for each time step t.

HER offers two strategies to infer an alternative goal. Firstly, the **final** strategy infers an alternative goal using the re-labelling/mapping function on the **final** state of the unsuccessful trajectory of length T, $\hat{g}^0 = m(s_T)$, and the corresponding rewards are computed via the learning reward function using the predicate oracle f, $\forall t \in [0, T-1], r^0_t = r_{learning}(s_t, a_t, s_{t+1}, \hat{g}^0) = f(s_{t+1}, \hat{g}^0)$. Or, any of the **future-k** strategies can be used, with $k \in \mathbb{N}$ being an hyperparameter. They consist of applying the **final** strategy to k different, contiguous sub-parts of the main trajectory.

2.3 HIGhER and its Limitations.

HIGhER[16] aims to expand the applicability of HER, and to do so it explores how to learn the re-labelling/mapping function (hereafter referred to as m_{HIGhER} and Instruction Generator), rather than assuming it is provided or by using some form of external expert knowledge. Nevertheless, it still relies on a predicate function being provided and queried as an oracle. HIGhER investigates using hindsight experience replay in the instruction following setting from pixel-based observations, which brings some particularities as it differs from the robotic setting of HER. Firstly, the goal space and state space are no longer the same, hence the motivation towards learning a re-labelling/mapping function. Secondly, there is no obvious mapping from stimuli (e.g. visual/pixel states) to the instructions that define the goals using a natural-like language. For instance, for a given state, due to the expressivity of natural languages, multiple goals may be defined as being fulfilled in this state.

Despite the non-obvious mapping from stimuli to fulfilled goals, HIGhER still succeeds in learning a deterministic re-labelling/mapping function. HIGhER learns an instruction generator by supervised learning on a dataset $\mathcal{D}_{sup} = \{(s, g)/f(s, g) = 1\}$ consisting of state-goal pairs where the predicate value is know to be 1. These pairs are harvested from successful trajectories of the RL agent which occur throughout the learning process (they could be provided as demonstrations, but this is not explored by the original work of Cideron et al. [16]) and correspond to final states s of successful/positive-reward trajectories along with relevant linguistic instructions defining the fulfilled


Dataset
D_{RG}
Target Stimulus
s_0
Speaker
Stimulus Description / Message m_0
STGS Communication Channel
Listener
Listener Decision
Loss
$L_{STGS-LazImpa}$
Distractor Stimuli
s_1 s_2
s_{K-1} s_K
Shuffle
: Data Augmentation Scheme
Listener Agent
MLP
Speaker Agent / Instruction Generator
MLP
Convolutional Feature Maps
ReLU
BN
Conv
ReLU
BN
Conv
ReLU
BN
Conv
Shared Observation Encoder
FiLM-ed Layers
Reinforcement Learning Agent
GRU
Stimulus (Stacked Consecutive Observations)
Linguistic Goal Description
"Pick up the purple ball"

(a) (b)

Figure 1: **(a):** Illustration of a *discriminative object-centric/2-players/L-signal/*$N = 0$*-round/K-distractor visual referential game* [18] using a Straight-Through Gumbel-Softmax (STGS) communication channel (following the approach of Havrylov and Titov [23]) and a loss function which adapts the STGS communication channel to LazImpa from Rita et al. [57], that we refer to as the STGS-LazImpa loss function (cf. Appendix E.1). Stimuli are passed through a data augmentation scheme, following the recipe from Dessi et al. [20] (i.e. adding Gaussian Blur, and/or Color Jitter, and/or undergoing some Affine Transformation) in order to enforce object-centricism [15, 18] Note that the target stimulus undergoes different data augmentation depending on whether it is fed to the speaker agent or the listener agent. **(b):** ETHER agent architecture describing the *Shared Observation Encoder* that feeds its output **convolutional feature maps** to both RG agents and the RL agent. Note that, prior to being fed to any agent, some form of agent-specific adaptations are applied to the feature maps, with the RL agent having the most sophisticated ones relying on some *FiLM-ed layers* [53] that are conditioned on the output of a *GRU layer* that embeds the *linguistic goal description*.

goals g. The ability to harvest successful trajectories from an RL agent in the process of being trained is capped by how likely is it that this RL agent will fulfill goals albeit while randomly/cluelessly exploring the environment. Thus, one major limitation of HIGhER is that in the absence of initial (and therefore random - when harvested from the learning RL agent) successful trajectories, the dataset $\mathcal{D}_{sup}$ cannot be built, and it ensues that the hindsight experience replay scheme cannot be leveraged since the instruction generator/mapping function, m_{HIGhER}, cannot begin to learn.

Finally, it is important to note that HIGhER is constrained to using only the **final** re-labelling strategy. Recall that the Instruction Generator is solely trained on episode's final state, and it is likely that the distribution of final states over the whole state space $\mathcal{S}$ is far from being uniform. Thus, applying the re-labelling/mapping function of HIGhER on states encountered in the middle of an episode is tantamount to out-of-distribution application and would likely result in unpredictable re-labelling mistakes. In the original work of Cideron et al. [16], this particularity is not addressed, and only the **final** re-labelling strategy is experimented with.

2.4 Emergent Communication

Emergent Communication is at the interface of language grounding and language emergence. While language emergence raises the question of how to make artificial languages emerge, possibly with similar properties to natural languages, such as compositionality [3, 21, 43, 56], language grounding is concerned with the ability to ground the meaning of (natural) language utterances into some sensory processes, e.g. the visual modality. On one hand, emergent artificial languages' compositionality has been shown to further the learnability of said languages [37, 60, 7, 43] and, on the other hand, natural languages' compositionality promises to increase the generalisation ability of the artificial agent that would be able to rely on them as a grounding signal, as it has been found to produce learned representations that generalise, when measured in terms of the data-efficiency of subsequent transfer and/or curriculum learning [25, 48, 49, 32]. Yet, emerging languages are far from being 'natural-like' protolanguages [39, 9, 10], and the questions of how to constraint them to a specific semantic or a specific syntax remain open problems. Nevertheless, some sufficient conditions can be found to further the emergence of compositional languages and generalising learned representations (e.g. Kottur et al. [39], Lazaridou et al. [41], Choi et al. [15], Bogin et al. [4], Guo et al. [21], Korbak et al. [38], Chaabouni et al. [11], Denamganaï and Walker [19]).

The backbone of the field rests on games that emphasise the functionality of languages, namely, the ability to efficiently communicate and coordinate between agents. The first instance of such an environment is the *Signaling Game* or *Referential Game (RG)* by Lewis [42], where a speaker agent is asked to send a message to the listener agent, based on the *state/stimulus* of the world that it observed. The listener agent then acts upon the observation of the message by choosing one of the *actions* available to it in order to perform the 'best' *action* given the observed *state* depending on the notion of 'best' *action* being defined by the interests common to both players. In RGs, typically, the listener action is to discriminate between a target stimulus, observed by the speaker and prompting its message generation, and some other distractor stimuli. The listener must discriminate correctly while relying solely on the speaker's message. The latter defined the discriminative variant, as opposed to the generative variant where the listener agent must reconstruct/generate the whole target stimulus (usually played with symbolic stimuli). Visual (discriminative) RGs have been shown to be well-suited for unsupervised representation learning, either by competing with state-of-the-art self-supervised learning approaches on upstream classification tasks [20], or because they have been found to further some forms of disentanglement [26, 34, 12, 44] in learned representations [71, 17]. Such properties can enable "better up-stream performance"[66], greater sample-efficiency, and some form of (systematic) generalization [47, 24, 61]. Indeed, disentanglement is thought to reflect the compositional structure of the world, thus disentangled learned representations ought to enable an agent wielding them to generalize along those lines. Thus, this paper aims to investigate visual discriminative RGs as auxiliary tasks for RL agents.

Visual Discriminative Referential Game Setup. Following the nomenclature proposed in Denamganaï and Walker [18], we will focus primarily on a *descriptive object-centric (partially-observable)* 2-*players/L* $= 10$-*signal/N* $= 0$-*round/K* $= 31$-*distractor* RG variant, as illustrated in Figure 1a.

As an object-centric RG, as opposed to stimulus-centric, the listener and speaker agents are not being presented with the same exact target stimuli. Rather, they are being presented with different *viewpoints* on the same target object shown in the target stimuli, where the word *viewpoint* ought to be understood in a large sense. Indeed, object-centrism is implemented by applying data augmentation schemes such as gaussian blur, color jitter, and affine transformations, as proposed in Dessi et al. [20]. Thus, the listener and speaker agents would be presented with different stimuli that nevertheless keeps the conceptual object being presented constant. This aspect was introduced by Choi et al. [15] (without it being of primary interest), where the pair of agents would literally be shown potentially the same 3D objects under different viewpoint, thus thinking of object-centrism as a *viewpoint* shift is historically relevant.

Concerning the communication channel, it is parameterised with a Straight-Through Gumbel-Softmax (STGS) estimator following the work of Havrylov and Titov [23]. The vocabulary V is fixed with 62 ungrounded symbols, plus two grounded symbol accounting for the *Start-of-Sentence* and *End-of-Sentence* semantic, thus $|V| = 64$. The maximum sentence length L is always equal to 10, thus placing our experiments in the context of an overcomplete communication channel whose capacity is far greater than the number of different meanings that the agents would encounter in our experiments [40].

In this paper, we will focus exclusively on STGS parameterisation, but many other could have been used (e.g. REINFORCE-based algorithms [70], quantization [8] and Obverter approaches [15, 4]). Indeed, the STGS approach supposedly allows a richer signal towards solving the credit assignment problem that language emergence poses, since the gradient can be backpropagated from the listener agent to the speaker agent.

3 Method

In the following, firstly, we detail our proposal to slightly extend HIGhER's applicability, in Section 3.1, and then, we detail our novel architecture, entitled ETHER, that addresses all the limitations inherent to the HIGhER's paradigm.

3.1 Extending HIGhER's Applicability

As explained in Section 2.3, HIGhER, like HER, still relies on a predicate function being provided and queried as an oracle. In the following, we propose to address this limitation in two ways. We first propose to derive a partial predicate function from the re-labelling function in the Section 3.1.1. Then, we show how to enhance the quality of the derived predicate function with a contrastive learning approach in Section 3.1.2.

3.1.1 Deriving a Predicate Function

Because HIGhER only implements the **final** relabelling strategy, we remark that the predicate function is only necessary in order to compute the output of the *learning reward function* when it is fed a state s_t from an unsuccessful episode and the relabelling goal $g' \in \mathcal{G}$, such that $g' = m(s_{t_{final}})$. Notably, this new goal g' could have also been reached in previous steps of the same trajectory, and if so, those transitions should also feature a positive reward like the final state of the episode. This can be achieved by applying m to all states in the trajectory and giving a positive reward *if and only if* the new goal for a given state matches that of the relabelled goal for the last state s_{last}. This procedure derives a predicate function from a re-labelling function. In the remainder of the paper, we will denote this extension as HIGhER$+$.

It is important to note that this procedure is not as sound as it could be because it makes the implicit and erroneous assumption that fulfilled goals are deterministic and unique for each state, whereas, firstly, the expressivity of natural language allows many different ways of expressing a similar semantic for a goal that would have been fulfilled in a given state (resulting in different theoretically-valid values for $m(s_t)$ and $m(s_{t_{final}})$) and, secondly, for any given state a distribution of fulfilled goals (with different semantics, not just synonymous expressions) could be defined. For instance, whenever a “pick up the blue ball” can be identified as being fulfilled in a given state, then the goal “pick up a blue object”, or “pick up a ball” are all as valid as the former.

3.1.2 Enhancing the Predicate Function with Contrastive Learning

Let us assume that we have access to a relabelling function, either learnt or given. Successful trajectories, those that yield a positive environment reward, are ground-truth indicators of a goal being satisfied on the last visited state. In contrast, the states in the trajectory leading up the goal-fulfilling state do not satisfy the same condition as the last state, as otherwise those transitions would receive a positive reward according to the environment’s goal-conditioned reward signal. We exploit this structure to use contrastive learning methods.

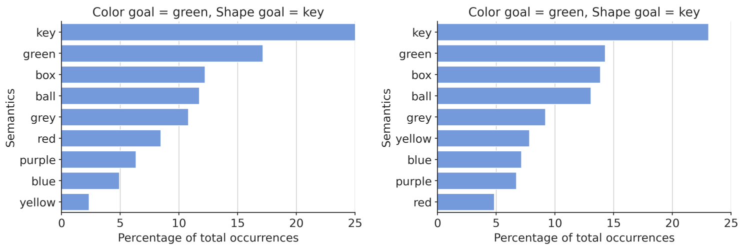
HIGhER learns an instruction relabelling function by making use of a dataset of (state, goal) pairs, as defined in Section 2. We further increase the accuracy of the learnt re-labelling function via contrastive learning where the positive examples are the same (state, goal) pairs in dataset $\mathcal{D}$ and the negative examples are defined as follows. Let T_{final} be the timestep of the final transition. Negative examples consist of pairs of states $S_{(T_{final}-i)}$ for $i \in [1, n]$ and their associated negative goal. This negative goal is built contrastively to the true re-labelling goals as $G_{neg} = EoS$, i.e. using the End of Sentence (EoS) symbol. We use EoS to trivially satisfy that the negative goal G_{neg} differs from the goal of the positive example g.

3.2 Leveraging Unsuccessful Trajectories with Emergent Communication

In the following, we rebase the architecture around the Emergent Communication paradigm in order to both learn a relabelling function, in the form of the speaker agent, and a predicate function, in the form of a listener agent. The resulting algorithm is illustrated in Figure 3. Figure 1b also highlights shared components between the RG agents and the RL agent in ETHER, thus allowing the RG to be acknowledged as an unsupervised auxiliary task for RL, following the work of Jaderberg et al. [30]. This change of paradigm brings about a new challenge in the alignment of the emergent language used by the RG agents to the natural-like language of the benchmark/environment. In the following Section 3.2.1, we highlight how to use an RG’s listener agent as a predicate function, and then we detail in Section 3.2.2 our proposal to align the emergent language with the environment’s language.

3.2.1 Learning a Predicate Function via Referential Games

Taking a closer look at the listener agent of a visual discriminative RG, it takes as input $K + 1$ stimuli and a message/linguistic description from the speaker agent to output confidence levels for each stimulus of the extent with which the message clearly describes them. In the context of $K = 0$, the listener agent outputs a likelihood for the message to be clearly describing some attributes wihtin



(a) Expert agent

(b) Random agent

Figure 2: Episode goal semantics (left columns) and count of observation semantics from trajectories conditioned on a given goal (right histogram). **Left:** trajectories from BabyAI's built-in expert agent which always reaches the goal. **Right:** random agent trajectories. In both cases the semantics of the goal are among the most observed semantic features for any given trajectory. This effect is less pronounced in the random agent.

the one and only stimulus provided. This is analoguous to what a predicate function does. Thus, the listener agent of any RG can be readily put in the place of the predicate function in the context of hindsight experience replay (as illustrated in Figure 3), provided that the RG's speaker agent is used as an Instruction Generator following HIGhER recipe. Note that this extension already incorporates contrastive learning as a discriminative RG is literally asking the listener agent to contrast positive (the target) and negative (the distractors) stimuli. Henceforth, we refer to this augmentation as the Emergent Textual Hindsight Experience Replay (ETHER) agent. We provide in Appendix E further details about the instantiated RG.

3.2.2 Aligning Emergent Languages via Semantic Co-occurrence Grounding

A major shortcoming of HIGhER is that during training it learns a goal relabelling function which is only capable of mapping states to goals that are satisfied in successful trajectories. Generally, these goals are represented through semantic descriptions of the necessary interactions between the agent and objects that are present in the environment (e.g., "pick up the green key", "open the door"). Presently, we hypothesise that if it is indeed the case that the goal can always be fulfilled, then, upon specifiying a goal, agent observations will be biased to contain semantic components present in said goal. We test this hypothesis in the BabyAI environment in Figure 2 where we see that, indeed, the semantics of the goal are some of the most salient observed semantics in both expert trajectories and random walks. Given the similarity between semantics of observations and goals in both successful and unsuccessful trajectories, which we refer to henceforth as **semantic co-occurrence**, we ask ourselves: How can this underlying environmental structure be leveraged to learn a semantic understanding of the goal trying to be achieved? In the context of ETHER which is centered around the addition of RG agents, this translates as: How can this underlying environmental structure be leveraged to constraint the RG's emergent language to be aligned with the natural(-like) language used to describe goals in any instruction-following benchmark?

To answer this question, we introduce the **semantic co-occurrence grounding loss**, which aims to enhance an agent's language grounding ability during RG training. We emphasise that this loss does not rely on private information from the environment, but solely makes use of what information an instruction-following agent can actually observe. To do so, only the words/tokens present in the linguistic goal description provided are used as labels. Formally, let us define a linguistic goal description as a series of tokens, $g = (g_i)_{i \in [1,L]} \in \mathcal{G}$, where L is the maximum sentence length hyperparameter, as defined in the RG setup (cf. Section 2.4 and Appendix E).

Thus, for each of those token present in the goal g of a given episode (out of all the tokens available in the vocabulary V, as defined in the RG setup), the semantic co-occurrence grounding loss will aim to bring a prior semantic-only embedding of the tokens closer to the visual embeddings of all the observations during the given episode. We will denote by $(\lambda_w)_{w \in V}$ all the prior semantic-only embeddings for the vocabulary V. And, on the other hand, it will also bring further away from the visual embeddings of all the observations during the episode the prior semantic-only embeddings of **all the tokens of the vocabulary that are not present in the current goal** g.

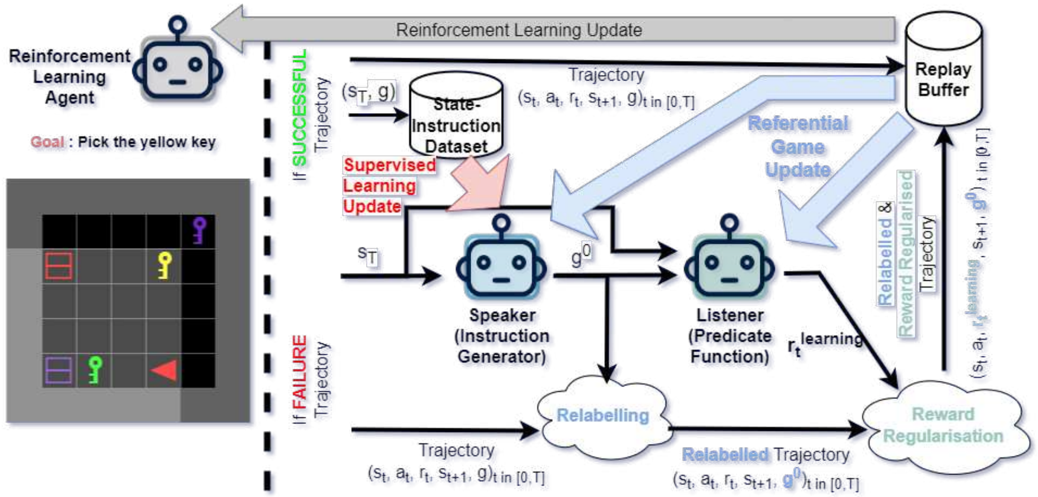

Reinforcement Learning Update
Reinforcement Learning Agent
Goal : Pick the yellow key
If SUCCESSFUL Trajectory
(s_T, g)
State-Instruction Dataset
Trajectory
(s_t, a_t, r_t, s_t+1, g)_t in [0,T]
Replay Buffer
Referential Game Update
Supervised Learning Update
s_T
g^0
Speaker (Instruction Generator)
Listener (Predicate Function)
r_t^learning
Relabelled & Reward Regularised Trajectory
(s_t, a_t, r_t^learning, s_t+1, g^0)_t in [0,T]
If FAILURE Trajectory
Trajectory
(s_t, a_t, r_t, s_t+1, g)_t in [0,T]
Relabelling
Relabelled Trajectory
(s_t, a_t, r_t, s_t+1, g^0)_t in [0,T]
Reward Regularisation

Figure 3: ETHER's algorithm relies on three agents, the two Referential Game (RG) agents, *speaker* and *listener*, and the Reinforcement Learning (RL) agent. When the RL agent generates a successful trajectory, effectively following the instruction described by the goal, the trajectory is added to the replay buffer and the final state s_T and episode's instruction g are added to the state-instruction dataset. Sampling from this dataset allow training of the *speaker* agent in a supervised learning fashion, effectively mimicking how the **instruction generator** from HIGhER is trained. On the other hand, when the RL agent generates a failed trajectory, the *speaker* agent is used as an **instruction generator** to relabel the trajectory, replacing the failed goal g with a linguistic description g^0 of the episode's final state s_T. Then, as this final state may be repeated throughout the episode, it is important to regularise the rewards throughout the trajectory. This is performed by using the *listener* agent as a **predicate function**. Finally, the relabelled and reward regularised trajectory can be added to the replay buffer. Sampling from the replay buffer will allow performing RL updates on the RL agent as well as RG updates on both the *speaker* and *listener* agents.

The semantic co-occurrence grounding loss is contrastive and inspired by Radford et al. [55]. More formally, as we defined $f(\cdot)$ as the visual module in Section 4.2, we write the semantic co-occurrence grounding loss as follows:

$$\mathcal{L}^{sem.}_{co-occ.\,ground}(g|(\lambda_w)_{w\in V}) = \mathbb{E}_{s\sim\rho^\pi}\left[\sum_{w\in V}\mathcal{H}(w)\sum_{g_i\in g}\left(\mathbf{1}_w(g_i) - \frac{\lambda_w\cdot f(s)^T}{||\lambda_w||_2\cdot||f(s)||_2}\right)^2\right], \quad (1)$$

where $||\cdot||_2$ corresponds to the $L2$ norm, ρ^π is the distribution over states in $s\in\mathcal{S}$ that is induced by using the policy π to harvest the observations/stimuli, and $1_w(\cdot)$ is a noisy indicator function defined as follows:

$$\mathbf{1}_w(w') := (1-\epsilon_{noise})\times\begin{cases}1 & \text{if } w' == w\ ,\\ -1 & \text{if } w'\neq w\ .\end{cases} \quad (2)$$

where ϵ_{noise} is some random noise uniformly sampled from $[0, 0.2]$, following the noisy labels idea proposed in Salimans et al. [58].

As the loss is implemented over mini-batches of sampled stimuli, we also perform masking to reject tokens with null entropy over the mini-batch. For instance, in the proposed experiments performed on BabyAI [13]'s PickupDist-v0 task, the linguistic goal description always contains the prefix 'pick up', therefore, when considering a mini-batch of stimuli (however they may come from different episodes), the likelihood of the tokens 'pick' and 'up' is maximal over the mini-batch and therefore their associated appearance distribution over the sampled stimuli will have null entropy. In Equation 1, $\mathcal{H}(w)$ denote the entropy of the appearance distribution of token $w\in V$, and its presence as a multiplicative term is for masking purposes. We refer to the architecture incorporating the loss of Equation 1 as ETHER+.

4 Experiments

4.1 Experimental setup

We perform all experiments in an altered version of the BabyAI environment 'BabyAI-PickUpDist-v0' [13]. This environment rewards an agent at each episode for picking up a specifically coloured and shaped object among other distracting objects, depending on an observed, natural(-like) language instruction, e.g. "Pick up the blue ball". We altered the environment by adding a one-pickup-per-episode wrapper that makes it so that the episode ends when any object is picked up, meaning that there is no pick-up action happening in the rest of the episode but the very last experience, unless the episode times out after 40 available timesteps (similarly to Cideron et al. [16]). In other words, the result of an episode can either be successful (the agent picked up the target object, as specified by the instruction), unsuccessful (the agent picked up a wrong object), or timed out (i.e. no object was picked-up within the limit of the 40 timesteps).

4.2 Agent Architecture

The ETHER architecture is made up of three differentiable agents, the language-conditioned RL agent and the two RG agents (speaker and listener). Similarly, the HIGhER architecture is built around a language-conditioned RL agent and an Instruction Generator, which plays the same role as the speaker agent in a RG thus we equate them in the following architectural details. Each agent consists of at least a language module and a visual module. The *listener* agent additionally incorporates a third decision module that combines the outputs of the other two modules. The RL agent similarly incorporates a third decision module with the addition that this third module contains a recurrent network, acting as core memory module for the agent. Using the Straight-Through Gumbel-Softmax (STGS) approach in the communication channel of the RG, the *speaker* agent is prompted to produce the output string of symbols with a *Start-of-Sentence* symbol and the visual module's output as an initial hidden state while the *listener* agent consumes the string of symbols with the null vector as the initial hidden state. In the following subsections, we detail each module architecture in depth.

Visual Module. The visual module $f(\cdot)$ consists of the *Shared Observation Encoder*, which is shared between all the different agents, followed by some agent-specific adaptation layers, as shown in Figure 1b. The former consists of three blocks of a 3×3 convolutional layer with stride 2 followed by a 2D batch normalization layer and a ReLU non-linear activation function. The two first convolutional layers have 32 filters, whilst the last one has 64. The bias parameters of the convolutional layers are not used, as it is common when using batch normalisation layers. Inputs are stimuli consisting of 4 stacked consecutive frames of the environment resized to 64×64. The RL agent's adaptation layers consist of 2 FiLM layers [53] conditioned on the *linguistic goal description* $g \in \mathcal{G}$ after it is processed by the RL agent's language module. Please refer to the appendix or the open-sourced code for the details on the other adaptation layers which consists of fully-connected networks.

Language Module. The language module $g(\cdot)$ consists of some learned Embedding followed by either a one-layer GRU network [14] in the case of the RL agent, or a one-layer LSTM network [27] in the case of the RG agents. In the context of the *listener* agent, the input message $m = (m_i)_{i \in [1,L]}$ (produced by the *speaker* agent) is represented as a string of one-hot encoded vectors of dimension $|V|$ and embedded in an embedding space of dimension 64 via a learned Embedding. The output of the *listener* agent's language module, $g^l(\cdot)$, is the last hidden state of the RNN layer, $h^l_L = g^L(m_L, h^l_{L-1})$. In the context of the *speaker* agent's language module $g^S(\cdot)$, the output is the message $m = (m_i)_{i \in [1,L]}$ consisting of one-hot encoded vectors of dimension $|V|$, which are sampled using the STGS approach from a categorical distribution $Cat(p_i)$ where $p_i = Softmax(\nu(h^s_i))$, provided ν is an affine transformation and $h^s_i = g^s(m_{i-1}, h^s_{i-1})$. $h^s_0 = f(s_t)$ is the output of the visual module, given the target stimulus s_t.

Decision Module. From the RL agent to the RG's listener agent, the decision module are very different since their outputs are either, respectively, in the action space $\mathcal{A}$ or the space of distributions over $K + 1$ stimuli. For the RL agent, the decision module takes as input a concatenated vector comprising the output of the FiLM layers, after it has been procesed by a 3-layer fully-connected network with 256, 128 and 64 hidden units with ReLU non-linear activation functions, and some other information relevant to the RL context (e.g. previous reward and previous action selected, following the recipe in Kapturowski et al. [33]). The resulting concatenated vector is then fed to the core memory module, a one-layer LSTM network [27] with 1024 hidden units, which feeds into the advantage and value heads of a 1-layer dueling network [68].

In the case of the RG's listener agent, similarly to Havrylov and Titov [23], the decision module builds a probability distribution over a set of $K+1$ stimuli/images $(s_0, ..., s_K)$, consisting of K distractor stimuli and the target stimulus, provided in a random order (see Figure 1a), given a message m using the scalar product:

$$p((d_i)_{i\in[0,K]}|(s_i)_{i\in[0,K]};m) = Softmax\Big((h_L^l \cdot f(s_i)^T)_{i\in[0,K]}\Big). \tag{3}$$

4.3 ETHER Improves Sample-Efficiency and Performance

Hypothesis. Firstly, our extensions to HIGhER replace the oracle predicate function with a derived, deterministic predicate function which is not theoretically sound, thus we investigate here whether this can still be beneficial to the RL agent's learning by comparison to our R2D2 baseline **(H1)**. Secondly, as ETHER instantiates hindsight learning and an unsupservised auxiliary task for RL in the form of the RG, we expect it to improve the sample-efficiency and the asymptotic performance compared to our R2D2 baseline **(H2)**. Thirdly, since ETHER learns a principled predicate function (in the form of the listener agent of a RG) which is theoretically sound, as opposed to the predicate functions derived in our various HIGhER extensions, we hypothesise that ETHER should also improve the sample-efficiency and the asymptotic performance compared to HIGhER extensions **(H3)**. Finally, as ETHER+ constraints the EL via semantic co-occurrence grounding, we ponder whether this constraint has any impact on the RL agent's performance **(H4)**.

Table 1: Success ratios (percentage of mean and standard deviation) for agents with burn-in feature of R2D2 after 200k observations.

Agent	Success Ratio
R2D2	16.54 ± 1.37
HIGhER+	14.84 ± 1.40
HIGhER++ (n=1)	15.89 ± 1.19
HIGhER++ (n=2)	16.80 ± 2.07
HIGhER++ (n=4)	18.10 ± 2.54
ETHER	27.63 ± 1.20
ETHER+	27.16 ± 2.57

Evaluation. We evaluate both the sample-efficiency and performance by reporting on the success ratio of the RL agents over 256 randomly-generated environments, after training has occured on a fixed sampling budget of 200k observations.

Results. Table 1 shows the success ratios for the different algorithms and architectures. We can see that our HIGhER extensions requires a great amount of negative examples in the contrastive training (n=4) in order to validate **(H1)**, and still it only provides marginal improvements over baseline. Thus, our results shows that our derived predicate function is practically feasable but fairly limited. On the otherhand, both ETHER approaches outperform all other approaches by almost doubling the final performance, thus validating both hypotheses **(H2)** and **(H3)**. These results are cementing the usage of visual discriminative referential games as viable unsupervised auxiliary task for RL and they are showing that our principled, RG-learned predicate function is not only theoretically sound but also practical. Finally, regarding **(H4)**, we observes similar mean asymptotic performance between ETHER and ETHER+, but ETHER+'s distribution has a greater standard deviation which goads us to think that the semantic co-occurrence grounding may exert some detrimental constraints onto the RL agent. We bring the reader's attention onto the fact that the noise parameter ϵ_{noise} of the semantic co-occurrence grounding loss (cf. equation 2) may still be too strong and thus explain this detrimental effect on the RL performance.

4.4 Semantic Co-Occurrence Grounding Improves Emergent Language Alignment

Hypothesis. Strong of the results showing similar RL performance betweem ETHER and ETHER+, we now investigate the alignment between the emergent languages wielded by the RG agents and the benchmark's natural-like language. We hypothesise that only ETHER+ provides some linguistic alignment because ETHER has no incentives to do so **(H1)**.

Evaluation. We propose two metrics, referred to as 'Any-Colour' and 'Any-Shape' accuracies, to report on the alignment between the emergent languages and the benchmark's natural language. Each metric is consistent with a different attribute of the objects encountered by the RL agent in the environment, to wit 'colour' and 'shape'. For the purpose of their computation, we use private information from the BabyAI benchmark that corresponds to the symbolic representation of the agent's field of view (as opposed to the pixel observation that the agents have as input state from the environment), here after referred to as symbolic image. The symbolic image describes the colour and shapes of the objects that are in the field of view of the agent using indices [1]. For each observation used in the RG training, we convert the indices of the corresponding symbolic image into colour and shape word tokens and check whether the RG's speaker agent use any of those word token in its emergent language description of the current observation. Thus, the 'Any-Colour' accuracy metric registers high accuracy for the current observation is and only if **any** of the visible object's colour-related word tokens is used, and vice versa with shape-related word tokens for the 'Any-Shape' accuracy metric. It is important to understand that these metrics only provides a lower bound to the true linguistic alignment between the emergent languages and the benchmark's natural language. Indeed, they do not allow verification of whether each word token of a given colour or shape are related to their expected semantic with a one-to-one/bijective relationship. Nevertheless, these metrics allow verification of whether the colour and shape information are being consistently used by the RG agents in ways that allows for interpretation of the emergent language utterances.

Results. Table 2 reports the 'Any-Colour' and 'Any-Shape' accuracies after 200k RL observations, showing the extent to which the RG's speaker agent has been using any of the natural language colour and shape word tokens to describe stimuli containing said colour or shape as visual feature. The results show that constraining of the RG agents using the semantic co-occurrence grounding loss in ETHER+ does start to provide alignment between the emergent language and the benchmark's natural-like language regarding the colour semantic alone (roughly 32% accuracy), as the shape semantic remains ungrounded (0% accuracy). Compared to ETHER's 9% of 'Any-Colour' accuracy, which is close to random performance on this metric, the results validate our hypothesis **(H1)**.

Table 2: Alignment accuracies (percentage of mean and standard deviation) between the emergent languages spoken by the RG's speaker agent and the benchmark's Natural Language, after training on 200k RL observations.

Agent	**Any-Colour**	**Any-Shape**
ETHER	9.117 ± 1.676	0.0 ± 0.0
ETHER+	32.75 ± 6.29	0.0 ± 0.0

While our proposed metrics here are limited, we highlight that we did also experiment with the conjunctions counterparts' metrics, the 'All-Colour' and 'All-Shape' metrics, as well as the 'Any-Object' and 'All-Object' metrics (checking whether any/all objects are mentioned, in terms of both their colour **and** their shape ; in order to disambiguate from the case where the emergent language description would make use of the colour-related token for a first visible object and the shape-related token to another visible object rather than the same as the colour-related token's one), but none of the architecture were able to perform better than 0% on these metrics. We plan to investigate in further details in future works how to make progress on these more difficult alignment metrics, as well as include metrics related to the structure of the language, such as compositionality as it has been shown to improve learning [32].

5 Conclusion

In this paper, we proposed the Emergent Textual Hindsight Experience Replay (ETHER) agent, which builds on HER [2] and HIGhER [16] by adressing their limitations. Firstly, a discriminative visual referential game, commonly studied in the subfield of Emergent Communication, is used as an unsupervised auxiliary task. Secondly, a semantic grounding scheme is employed to align the emergent language with the natural language of the instruction-following benchmark. We show that the speaker and listener agents of the referential game make an artificial language emerge that can be aligned with the natural-like language used to describe goals in the BabyAI benchmark, and that it is expressive enough so as to also describe unsuccessful RL trajectories and thus provide feedback to the RL agent to leverage the linguistic, structured information contained in all trajectories. Our work shows that emergent communication is a viable unsupervised auxiliary task for goal-conditioned RL in sparse reward settings and provides missing pieces, i.e. a learned predicate function, to make HER more widely applicable.

[1] cf. `https://minigrid.farama.org/api/wrappers/#symbolic-obs`

Acknowledgments and Disclosure of Funding

This work was supported by the EPSRC Centre for Doctoral Training in Intelligent Games & Games Intelligence (IGGI) [EP/L015846/1].

We gratefully acknowledge the use of Python[65], IPython[54], SciPy[67], Scikit-learn[52], Scikit-image[64], NumPy[22], Pandas[69, 50], OpenCV[6], PyTorch[51], TensorboardX[29], Tensorboard from the Tensorflow ecosystem[1], without which this work would not be possible.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[2] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

[3] M. Baroni. Linguistic generalization and compositionality in modern artificial neural networks. mar 2019. URL `http://arxiv.org/abs/1904.00157`.

[4] B. Bogin, M. Geva, and J. Berant. Emergence of Communication in an Interactive World with Consistent Speakers. sep 2018. URL `http://arxiv.org/abs/1809.00549`.

[5] D. Bouchacourt and M. Baroni. How agents see things: On visual representations in an emergent language game. aug 2018. URL `http://arxiv.org/abs/1808.10696`.

[6] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[7] H. Brighton. Compositional syntax from cultural transmission. *MIT Press*, Artificial, 2002. URL `https://www.mitpressjournals.org/doi/abs/10.1162/106454602753694756`.

[8] B. Carmeli, R. Meir, and Y. Belinkov. Emergent quantized communication. *arXiv preprint arXiv:2211.02412*, 2022.

[9] R. Chaabouni, E. Kharitonov, E. Dupoux, and M. Baroni. Anti-efficient encoding in emergent communication. *NeurIPS*, may 2019. URL `http://arxiv.org/abs/1905.12561`.

[10] R. Chaabouni, E. Kharitonov, A. Lazaric, E. Dupoux, and M. Baroni. Word-order biases in deep-agent emergent communication. may 2019. URL `http://arxiv.org/abs/1905.12330`.

[11] R. Chaabouni, E. Kharitonov, D. Bouchacourt, E. Dupoux, and M. Baroni. Compositionality and Generalization in Emergent Languages. apr 2020. URL `http://arxiv.org/abs/2004.09124`.

[12] R. T. Q. Chen, X. Li, R. Grosse, and D. Duvenaud. Isolating sources of disentanglement in VAEs. `https://papers.nips.cc/paper/2018/file/1ee3dfcd8a0645a25a35977997223d22-Paper.pdf`. Accessed: 2021-3-17.

[13] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio. BabyAI: First Steps Towards Grounded Language Learning With a Human In the Loop. oct 2018. URL `http://arxiv.org/abs/1810.08272`.

[14] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[15] E. Choi, A. Lazaridou, and N. de Freitas. Compositional Obverter Communication Learning From Raw Visual Input. apr 2018. URL `http://arxiv.org/abs/1804.02341`.

[16] G. Cideron, M. Seurin, F. Strub, and O. Pietquin. Higher: Improving instruction following with hindsight generation for experience replay. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 225–232. IEEE, 2020.

[17] K. Denamganaï, S. Missaoui, and J. A. Walker. Visual referential games further the emergence of disentangled representations. *arXiv preprint arXiv:2304.14511*, 2023.

[18] K. Denamganaï and J. A. Walker. Referentialgym: A framework for language emergence & grounding in (visual) referential games. *4th NeurIPS Workshop on Emergent Communication*, 2020.

[19] K. Denamganaï and J. A. Walker. On (emergent) systematic generalisation and compositionality in visual referential games with straight-through gumbel-softmax estimator. *4th NeurIPS Workshop on Emergent Communication*, 2020.

[20] R. Dessi, E. Kharitonov, and M. Baroni. Interpretable agent communication from scratch (with a generic visual processor emerging on the side). May 2021.

[21] S. Guo, Y. Ren, S. Havrylov, S. Frank, I. Titov, and K. Smith. The emergence of compositional languages for numeric concepts through iterated learning in neural agents. *arXiv preprint arXiv:1910.05291*, 2019.

[22] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020. doi: 10.1038/s41586-020-2649-2.

[23] S. Havrylov and I. Titov. Emergence of Language with Multi-agent Games: Learning to Communicate with Sequences of Symbols. may 2017. URL `http://arxiv.org/abs/1705.11192`.

[24] I. Higgins, A. Pal, A. Rusu, L. Matthey, C. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner. DARLA: Improving Zero-Shot Transfer in Reinforcement Learning. URL `https://arxiv.org/pdf/1707.08475.pdf`.

[25] I. Higgins, N. Sonnerat, L. Matthey, A. Pal, C. P. Burgess, M. Botvinick, D. Hassabis, and A. Lerchner. SCAN: Learning Abstract Hierarchical Compositional Visual Concepts. jul 2017. URL `http://arxiv.org/abs/1707.03389`.

[26] I. Higgins, D. Amos, D. Pfau, S. Racaniere, L. Matthey, D. Rezende, and A. Lerchner. Towards a Definition of Disentangled Representations. dec 2018. URL `http://arxiv.org/abs/1812.02230`.

[27] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[28] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

[29] T.-W. Huang. Tensorboardx, 2018. URL `https://github.com/lanpa/tensorboardX`.

[30] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, 2016.

[31] R. Jakobson. Linguistics and poetics. In *Style in language*, pages 350–377. MA: MIT Press, 1960.

[32] Y. Jiang, S. Gu, K. Murphy, and C. Finn. Language as an Abstraction for Hierarchical Deep Reinforcement Learning. jun 2019. URL `http://arxiv.org/abs/1906.07343`.

[33] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

[34] H. Kim and A. Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

[35] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[36] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[37] S. Kirby. Learning, bottlenecks and the evolution of recursive syntax. 2002.

[38] T. Korbak, J. Zubek, Ł. Kuciński, P. Miłoś, and J. Rączaszek-Leonardi. Developmentally motivated emergence of compositional communication via template transfer. oct 2019. URL `http://arxiv.org/abs/1910.06079`.

[39] S. Kottur, J. M. F. Moura, S. Lee, and D. Batra. Natural Language Does Not Emerge 'Naturally' in Multi-Agent Dialog. jun 2017. URL `http://arxiv.org/abs/1706.08502`.

[40] S. Kottur, J. M. F. Moura, D. Parikh, D. Batra, and M. Rohrbach. Visual Coreference Resolution in Visual Dialog using Neural Module Networks. sep 2018. URL `http://arxiv.org/abs/1809.01816`.

[41] A. Lazaridou, K. M. Hermann, K. Tuyls, and S. Clark. Emergence of Linguistic Communication from Referential Games with Symbolic and Pixel Input. apr 2018. URL `http://arxiv.org/abs/1804.03984`.

[42] D. Lewis. Convention: A philosophical study. 1969.

[43] F. Li and M. Bowling. Ease-of-Teaching and Language Structure from Emergent Communication. jun 2019. URL `http://arxiv.org/abs/1906.02403`.

[44] F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem. A sober look at the unsupervised learning of disentangled representations and their evaluation. Oct. 2020.

[45] J. Luketina, N. Nardelli, G. Farquhar, J. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel. A survey of reinforcement learning informed by natural language, 2019.

[46] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL `http://arxiv.org/abs/1312.5602`.

[47] M. L. Montero, C. J. Ludwig, R. P. Costa, G. Malhotra, and J. Bowers. The role of disentanglement in generalisation. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=qbH974jKUVy`.

[48] I. Mordatch and P. Abbeel. Emergence of Grounded Compositional Language in Multi-Agent Populations. URL `https://arxiv.org/pdf/1703.04908.pdf`.

[49] K. Moritz Hermann, F. Hill, S. Green, F. Wang, R. Faulkner, H. Soyer, D. Szepesvari, W. M. Czarnecki, M. Jaderberg, D. Teplyashin, M. Wainwright, C. Apps, D. Hassabis, P. Blunsom, and D. London. Grounded Language Learning in a Simulated 3D World. URL `https://arxiv.org/pdf/1706.06551.pdf`.

[50] T. pandas development team. pandas-dev/pandas: Pandas, Feb. 2020. URL `https://doi.org/10.5281/zenodo.3509134`.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011. URL `http://jmlr.org/papers/v12/pedregosa11a.html`.

[53] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[54] F. Perez and B. E. Granger. Ipython: A system for interactive scientific computing. *Computing in Science Engineering*, 9(3):21–29, 2007. doi: 10.1109/MCSE.2007.53.

[55] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[56] Y. Ren, S. Guo, M. Labeau, S. B. Cohen, and S. Kirby. Compositional Languages Emerge in a Neural Iterated Learning Model. feb 2020. URL `http://arxiv.org/abs/2002.01365`.

[57] M. Rita, R. Chaabouni, and E. Dupoux. " lazimpa": Lazy and impatient neural agents learn to communicate efficiently. *arXiv preprint arXiv:2010.01878*, 2020.

[58] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[59] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.

[60] K. Smith, S. Kirby, H. B. A. Life, and U. 2003. Iterated learning: A framework for the emergence of language. *Artificial Life*, 9(4):371–389, 2003. URL `https://www.mitpressjournals.org/doi/abs/10.1162/106454603322694825`.

[61] X. Steenbrugge, S. Leroux, T. Verbelen, and B. Dhoedt. Improving generalization for abstract reasoning tasks using disentangled feature representations. Nov. 2018.

[62] U. Strauss, P. Grzybek, and G. Altmann. *Word length and word frequency*. Springer, 2007.

[63] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[64] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[65] G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

[66] S. van Steenkiste, F. Locatello, J. Schmidhuber, and O. Bachem. Are disentangled representations helpful for abstract visual reasoning? May 2019.

[67] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[68] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[69] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. doi: 10.25080/Majora-92bf1922-00a.

[70] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[71] Z. Xu, M. Niethammer, and C. Raffel. Compositional generalization in unsupervised compositional representation learning: A study on disentanglement and emergent language. Oct. 2022.

[72] G. K. Zipf. *Human behavior and the principle of least effort: An introduction to human ecology*. Ravenio Books, 2016.

A Broader impact

No technology is safe from being used for malicious purposes, which equally applies to our research. However, we view many of the ethical concerns surrounding research to be mitigated in the present case. These include data-related concerns such as fair use or issues surrounding use of human subjects, given that our data consists solely of simulations.

With regards to the ethical aspects related to its inclusion in the field of Artificial Intelligence, we argue that our work aims to have positive outcomes on the development of human-machine interfaces since we investigate, among other things, alignment of emergent languages with natural-like languages.

The current state of our work does not allow extrapolation towards negative outcomes. We believe that this work is of benefit to the research community of reinforcement learning, language emergence and grounding, in their current state.

B Implementation Details

Table 3 highlights the hyperparameters used for the off-policy RL algorithm, R2D2[33]. More details can be found, for reproducibility purposes, in our open-source implementation at `https://github.com/Near32/Regym/tree/develop-ETHER/benchmark/ETHER`.

Training was performed for each run on 1 NVIDIA GTX1080 Ti, and the amount of training time for a run is between 8 and 24 hours depending on the architecture.

Table 3: Hyper-parameter values relevant to R2D2 in the different architectures presented. All missing parameters follow the ones in Ape-X [28].

R2D2	
Number of actors	32
Actor update interval	1 env. step
Sequence unroll length	20
Sequence length overlap	10
Sequence burn-in length	10
N-steps return	3
Replay buffer size	1×10^4 obs.
Priority exponent	0.9
Importance sampling exponent	0.6
Discount γ	0.98
Minibatch size	64
Optimizer	Adam [35]
Learning rate	6.25×10^{-5}
Adam ϵ	10^{-12}
Target network update interval	2500 updates
Value function rescaling	None

C On HIGhER's Ablation Study

Prior to the architectures described in the main part of the paper, we iterated over many designs and induction biases. Notably we experimented with R2D2's burn-in feature.

Table 4 shows the success ratios of HIGhER agents without burn-in feature against baseline R2D2 without burn-in feature on the modified (one-pick-up) PickUpDist-v0 task from the BabyAI benchmark at the end of the $200k$ observation budget. The results show that the contrastive learning scheme for the predicate function is rather hurting performance compared to HIGhER+, while still being above baseline. The burn-in feature provides the RL agent better sample-efficiency by stabilising the training of the recurrent network in the architecture. While the instruction generator/speaker agent is being trained, the resulting goal re-labelled experiences that enters the replay buffer are presumably non-stationary. Thus, we attribute the lower performance of the above architectures to the fact that they struggle to deal with the non-stationarity of the goal re-labelled experiences in the absence of the stabilising burn-in feature.

Table 4: Success ratios (mean and standard deviation) for agents without the burn-in feature of R2D2 after 200k steps in a modified version of the BabyAI PickUpDist-v0 task. 3 random seeds for each agent.

Agent	**Mean**
R2D2 (w/o Burn-In)	13.02 ± 1.26
HIGhER+ (w/o Burn-In)	16.02 ± 1.79
HIGhER++ (n=1) (w/o Burn-In)	14.97 ± 1.19
HIGhER++ (n=2) (w/o Burn-In)	15.89 ± 0.60
HIGhER++ (n=4) (w/o Burn-In)	13.93 ± 2.29

D On algorithmic details of ETHER

In this section, we detail how ETHER is built over HIGhER from an algorithmic point of view. We start by presenting in Algorithm 1 an extended version of the pseudo-code for the HIGhER algorithm from Cideron et al. [16] with the following additions:

1. (i) contrasting further the **learning** vs **behavioural** reward function concerns that we highlighted in Section 2.2,
2. (ii) flagging the reliance on the **learning** reward function that depends on the predicate function, which is provided as an oracle in both HER and HIGhER.

Algorithm 1: Hindsight Generation for Experience Replay (HIGhER)

```
Given :
    • an off-policy RL algorithm (e.g., DQN, R2D2) and its replay buffer $R$,
    • a behavioural policy $\pi_{behaviour} : \mathcal{S} \times \mathcal{G} \rightarrow \mathcal{A}$
    • a learning reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{G} \rightarrow \mathbb{R}$ (oracle or learned - relying on the
      predicate function $f : \mathcal{S} \times \mathcal{G} \rightarrow \{0, 1\}$),
    • a behavioural reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{G} \rightarrow \mathbb{R}$ (provided by the environment),
    • a language scoring function (e.g., parser accuracy, BLEU, etc.).
Initialize :
    • the dataset $\mathcal{D}_{sup}$ of $(state, goal)$ pairs and a train-test split strategy to yield $\mathcal{D}_{sup/train}$ and
      $\mathcal{D}_{sup/val}$,
    • the Instruction Generator $m_{HIGhER}$.
for $episode = 1, M$ do
    Sample a goal $g$ and an initial state $s_0$ from the environment;
    $t = -1$;
    repeat
        $t = t + 1$;
        Execute an action $a_t$ chosen from the behavioural policy $\pi_{behavioural}$;
        Observe a new state $s_{t+1}$ and a behavioural reward $r_t = r_{behavioural}(s_t, a_t, g)$;
        Store the transition $(s_t, a_t, r_t, s_{t+1}, g)$ in $R$;
        Update Q-network parameters using the policy $\pi_{behavioural}$ and sampled minibatches
          from $R$;
    until (episode ends);
    if learning reward $r_{learning}(s_t, a_t, s_{t+1}, g) = f(s_{t+1}, g) == 1$ then
        Store the pair $(s_{t+1}, g)$ in $\mathcal{D}_{sup/train}$ or $\mathcal{D}_{sup/val}$;
        Update $m_{HIGhER}$ parameters by sampling minibatches from $\mathcal{D}_{sup/train}$;
    end
    else if $m_{HIGhER}$ validation score is high enough & $\mathcal{D}_{sup/val}$ is big enough then
        Duplicate the previous episode's transitions in $R$;
        Sample $\hat{g}^0 = m_{HIGhER}(s_{t+1})$;
        Compute the learning rewards $\forall t, \hat{r}_t^0 = r_{learning}(s_t, a_t, s_{t+1}, \hat{g}^0) = f(s_{t+1}, \hat{g}^0)$;
        Replace $g$ by $\hat{g}^0$ and $r_t$ by $\hat{r}_t^0$ in all the duplicated transitions of the last episode;
    end
end
```

Following the added nuances to the HIGhER algorithm, we can now show in greater and contrastive details the ETHER algorithm in Algorithm 2, where we highlight the following:

1. (i) the RG training can be done in parallel at any time, thus we present it in the most-inner loop of the algorithm,
2. (ii) since ETHER trains its RG speaker and listener agents on the whole state space $\mathcal{S}$, the ability to perform either **final** or **future-k** re-labelling strategy is recovered. We present the case of the **future-k** re-labelling strategy below.

Algorithm 2: Emergent Textual Hindsight Experience Replay (ETHER)

```
Given :
    • an off-policy RL algorithm (e.g., DQN, R2D2) and its replay buffer $R$,
    • a behavioural policy $\pi_{behaviour} : \mathcal{S} \times \mathcal{G} \rightarrow \mathcal{A}$
    • a descriptive, discriminative RG algorithm, with its dataset buffer $\mathcal{D}_{RG}$ and its listener and speaker agents;
    • a learning reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{G} \rightarrow \mathbb{R}$ ( relying on the predicate function $f : \mathcal{S} \times \mathcal{G} \rightarrow \{0, 1\}$ which is implemented via the RG's listener agent),
    • a behavioural reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{G} \rightarrow \mathbb{R}$ (provided by the environment),
    • a language scoring function (implemented via the RG's accuracy on the validation set).
Initialize :
    • the dataset $\mathcal{D}_{sup}$ of $(state, goal)$ pairs and a train-test split strategy to yield $\mathcal{D}_{sup/train}$ and $\mathcal{D}_{sup/val}$,
    • the RG dataset $\mathcal{D}_{RG}$ of stimuli $state$ and a train-test split strategy to yield $\mathcal{D}_{RG/train}$ and $\mathcal{D}_{RG/val}$,
    • the Instruction Generator $m_{ETHER}(\cdot)$, in the form of the RG's speaker agent.
    • the learned predicate function $f_{ETHER}(\cdot)$, in the form of the RG's listener agent,
    • $K_{HER} \in \mathbb{N}$ specifying which re-labelling strategy to use (if $K_{HER} = 0$ then final, otherwise future-$K_{HER}$).
for $episode = 1, M$ do
    Sample a goal $g$ and an initial state $s_0$ from the environment;
    $t = -1$;
    repeat
        $t = t + 1$;
        Execute an action $a_t$ chosen from the behavioural policy $\pi_{behavioural}$;
        Observe a new state $s_{t+1}$ and a behavioural reward $r_t = r_{behavioural}(s_t, a_t, g)$;
        Store the transition $(s_t, a_t, r_t, s_{t+1}, g)$ in $R$;
        Update Q-network parameters using the policy $\pi_{behavioural}$ and sampled minibatches
          from $R$;
        Store the stimulus $s_t$ in $\mathcal{D}_{RG/train}$ or $\mathcal{D}_{RG/val}$;
        Update the RG's speaker and listener agents by playing $N_{RG}$ epochs of the RG, training
        on $\mathcal{D}_{RG/train}$ and performing evaluation on $\mathcal{D}_{RG/val}$;
    until episode ends;
    if learning reward $r_{learning}(s_t, a_t, s_{t+1}, g) = f_{ETHER}(s_{t+1}, g) == 1$ then
        Store the pair $(s_{t+1}, g)$ in $\mathcal{D}_{sup/train}$ or $\mathcal{D}_{sup/val}$;
        Update the RG's speaker agent parameters (ETHER) with supervised learning by
          sampling minibatches from $\mathcal{D}_{sup/train}$;
    end
    else if RG validation accuracy on $\mathcal{D}_{RG/val}$ is high enough then
        Use the future-$K_{HER}$ re-labelling strategy as follows...;
        $k = 0, T =$ last episode's length;
        repeat
            Sample $T_k$ uniformly from $[1, T]$;
            Duplicate the previous episode's transitions in $R$, until sampled timestep $T_k$;
            Sample $\hat{g}^0 = m_{ETHER}(s_{T_k})$;
            Compute the learning rewards
              $\forall t, \hat{r}^0_t = r_{learning}(s_t, a_t, s_{t+1}, \hat{g}^0) = f_{ETHER}(s_{t+1}, \hat{g}^0)$;
            Replace $g$ by $\hat{g}^0$ and $r_t$ by $\hat{r}^0_t$ in all the duplicated transitions of the last episode;
            $k = k + 1$;
        until $k == K_{HER}$;
    end
end
```

E On the Referential Game in ETHER

In the following, we detail further the referential game (RG) used in the ETHER architectures.

As highlighted in Section 2.4, we follow the nomenclature proposed in Denamganaï and Walker [18] and focus on a *descriptive object-centric (partially-observable)* 2*-players/*$L = 10$*-signal/*$N = 0$*-round/*$K = 31$*-distractor* RG variant, as illustrated in Figure 1a.

The descriptiveness implies that the target stimulus may not be passed to the listener agent, but instead replaced with a descriptive distractor. In effect, the listener agent's decision module therefore outpus a $K+2$-logit distribution where the $K+2$-th logit represents the meaning/prediction that none of the $K+1$ stimuli is the target stimulus that the speaker agent was 'talking' about. The addition is made following Denamganaï et al. [17] as a learnable logit value, $logit_{no-target}$, it is an extra parameter of the model. In this case the decision module output is no longer as specified in Equation 3, but rather as follows:

$$p((d_i)_{i\in[0,K+1]}|(s_i)_{i\in[0,K]};m) = Softmax\Big((h^l_L \cdot f(s_i)^T)_{i\in[0,K]} \cup \{logit_{no-target}\}\Big). \quad (4)$$

The descriptiveness is ideal but not necessary in order to employ the listener agent as a predicate function for the hindsight experience replay scheme. Thus, in the main results of the paper, we present the version without descriptiveness.

In the remainder of this section, we detail the STGS-LazImpa loss that we employed in our referential game, as illustrated in Figure 1a.

E.1 STGS-LazImpa Loss

Emergent languages rarely bears the core properties of natural languages [39, 5, 41, 11], such as Zipf's law of Abbreviation (ZLA). In the context of natural languages, this is an empirical law which states that the more frequent a word is, the shorter it tends to be [72, 62]. Rita et al. [57] proposed LazImpa in order to make emergent languages follow ZLA.

To do so, Lazimpa adds to the speaker and listener agents some constraints to make the speaker lazy and the listener impatient. Thus, denoting those constraints as $\mathcal{L}_{STGS-lazy}$ and $\mathcal{L}_{impatient}$, we obtain the STGS-LazImpa loss as follows:

$$\mathcal{L}_{STGS-LazImpa}(m,(s_i)_{i\in[0,K]}) = \mathcal{L}_{STGS-lazy}(m) + \mathcal{L}_{impatient}(m,(s_i)_{i\in[0,K]}). \quad (5)$$

In the following, we detail those two constraints.

Lazy Speaker. The Lazy Speaker agent has the same architecture as common speakers. The 'Laziness' is originally implemented as a cost on the length of the message m directly applied to the loss, of the following form:

$$\mathcal{L}_{lazy}(m) = \alpha(acc)|m| \quad (6)$$

where acc represents the current accuracy estimates of the referential games being played, and α is a scheduling function, which is not differentiable. This is aimed to adaptively penalize depending on the message length. Since the lazyness loss is not differentiable, they ought to employ a REINFORCE-based algorithm for the purpose of credit assignement of the speaker agent.

In this work, we use the STGS communication channel, which has been shown to be more sample-efficient than REINFORCE-based algorithms [23], but it requires the loss functions to be differentiable. Therefore, we modify the lazyness loss by taking inspiration from the variational autoencoders (VAE) literature [36].

The length of the speaker's message is controlled by the appearance of the EoS token, wherever it appears during the message generation process that is where the message is complete and its length is fixed. Symbols of the message at each position are sampled from a distribution over all the tokens in the vocabulary that the listener agent outputs. Let (W_l) be this distribution over all tokens $w \in V$ at position $l \in [1,L]$, such that $\forall l \in [1,L]$, $m_l \sim (W_l)$. We devise the lazyness loss as a Kullbach-Leibler divergence $D_{KL}(\cdot|\cdot)$ between these distribution and the distribution (W_{EoS})

which attributes all its weight on the EoS token. Thus, we dissuade the listener agent from outputting distributions over tokens that deviate too much from the EoS-focused distribution (W_{EoS}), at each position l with varying coefficients $\beta(l)$. The coefficient function $\beta : [1, L] \to \mathbb{R}$ must be monotically increasing. We obtain our STGS-lazyness loss as follows:

$$\mathcal{L}_{STGS-lazy}(m) = \sum_{l \in [1,L]} \beta(l) D_{KL}\Big((W_{EoS})|(W_l)\Big) \tag{7}$$

Impatient Listener. Our implementation of the Impatient Listener agent follows the original work of Rita et al. [57]: it is designed to guess the target stimulus as soon as possible, rather than solely upon reading the EoS token at the end of the speaker's message m. Thus, following Equation 3, the Impatient Listener agent outputs a probability distribution over a set of $K + 1$ stimuli $(s_0, ..., s_K)$ for all sub-parts/prefixes of the message $m = (m_1, ..., m_l)_{l \in [1,L]} = (m_{\leq l})_{l \in [1,L]}$:

$$\forall l \in [1, L], \;\; p((\mathbf{d_i^{\leq l}})_{\mathbf{i \in [0,K]}}|(s_i)_{i \in [0,K]}; \mathbf{m^{\leq l}}) = Softmax\Big((\mathbf{h_{\leq l}} \cdot f(s_i)^T)_{i \in [0,K]}\Big), \tag{8}$$

where $\mathbf{h_{<l}}$ is the hidden state/output of the recurrent network in the language module (cf. Section 4.2) after consuming tokens of the message from position 1 to position l included.

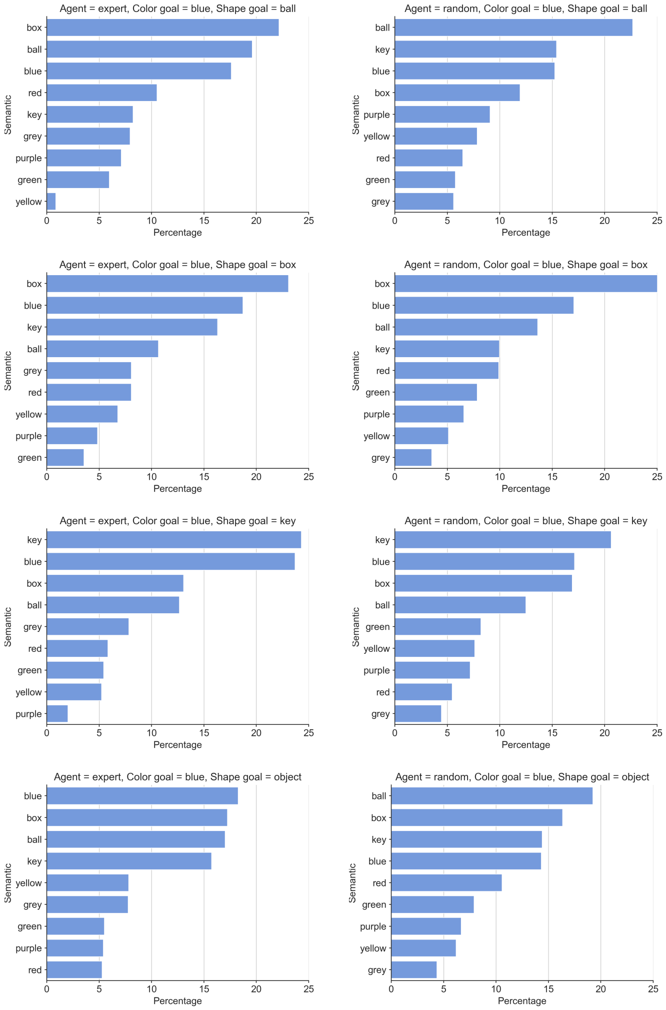
Thus, we obtain a sequence of L probability distributions, which can each be contrasted, using the loss of the user's choice, against the target distribution (D_{target}) attributing all its weights on the decision d_{target} where the target stimulus was presented to the listener agent. Here, we employ Havrylov and Titov [23]'s Hinge loss. Denoting it as $\mathbb{L}(\cdot)$, we obtain the impatient loss as follows:

$$\mathcal{L}_{impatient/\mathbb{L}}(m, (s_i)_{i \in [0,K]}) = \frac{1}{L} \sum_{l \in [1,L]} \mathbb{L}((d^{\leq l}_{i \in [0,K]}, (D_{target})). \tag{9}$$

F On the Semantic Co-Occurrence Hypothesis

In Section 3.2.2, we hypothesised that, upon specifiying a goal, agent observations would be biased to contain semantic components present in said goal. We tested this hypothesis in the BabyAI environment and provided some examples in Figure 2, when the linguistic goal description was "Pick up the green key", showing that the semantics of the goal (colour "green" and shape "key") are some of the most salient observed semantics in the environment's observations of both expert trajectories and random walks.

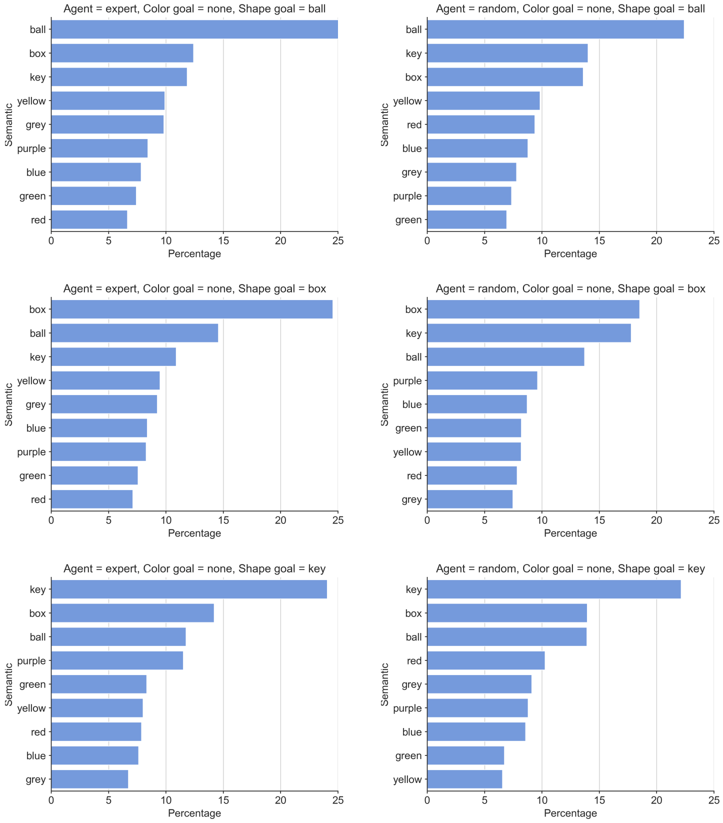
Here, we provide further evidence that the linguistic goal description aligns with the observed semantics across different permutations of color goal and shape goal. As described in the main body text, this is consistent across both agents, but more visible in the expert agent. Figures 5, 6, 4, 7, 8, 9, and 10 present histograms for each combination of color and shape goal for both the expert and random agent. We note that semantic co-occurrence, while prevalent, is not always perfectly the case. For instance, Figure 4, the most commonly observed semantic in the expert agent trajectories for the blue color and ball shape was "box", as opposed to the expected "ball" semantic.



(a) Expert agent

(b) Random agent

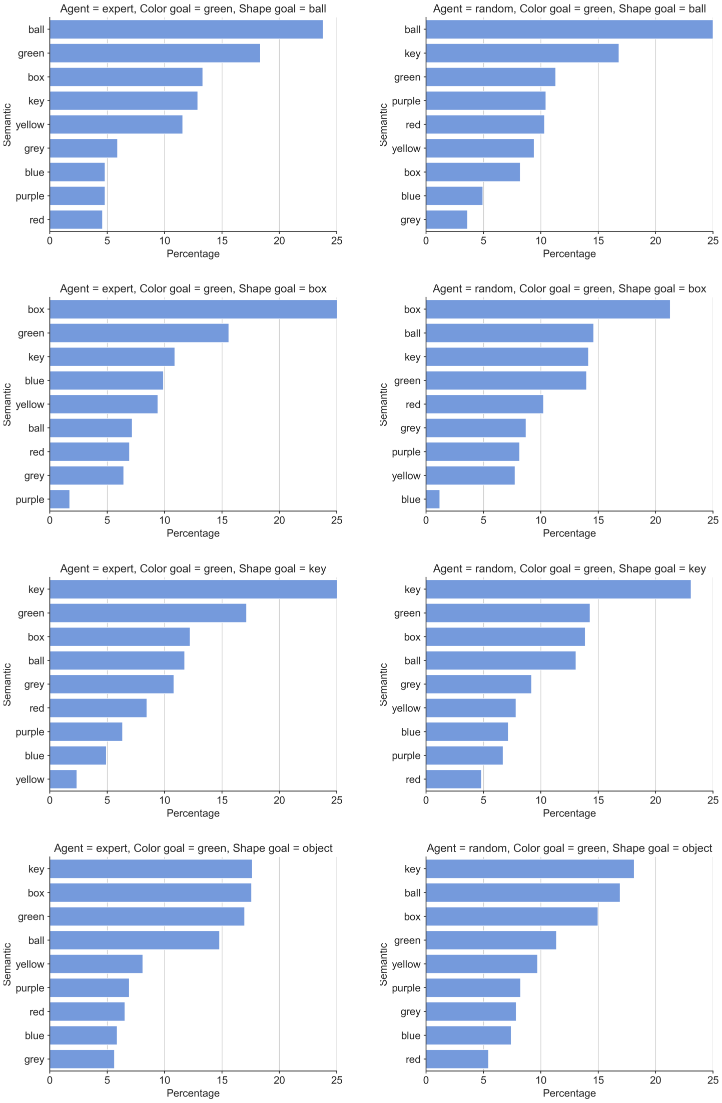
Figure 4: **Left:** Trajectories for the blue color goal from BabyAI's built-in expert agent which always reaches the goal. **Right:** Random agent trajectories. In both cases the semantics of the goal are among the most observed semantic features for any given trajectory. This effect is less pronounced in the random agent.



(a) Expert agent

(b) Random agent

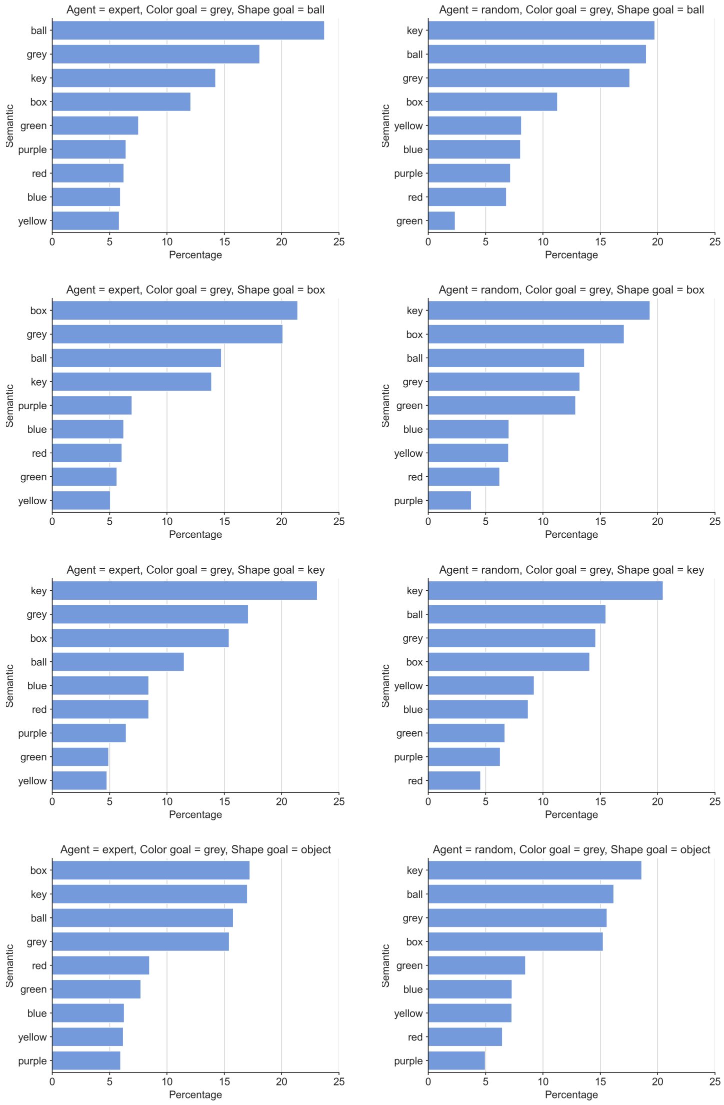
Figure 5: **Left:** Trajectories for colorless obejcts from BabyAI's built-in expert agent which always reaches the goal. **Right:** Random agent trajectories. In both cases the semantics of the goal are among the most observed semantic features for any given trajectory. This effect is less pronounced in the random agent.



(a) Expert agent

(b) Random agent

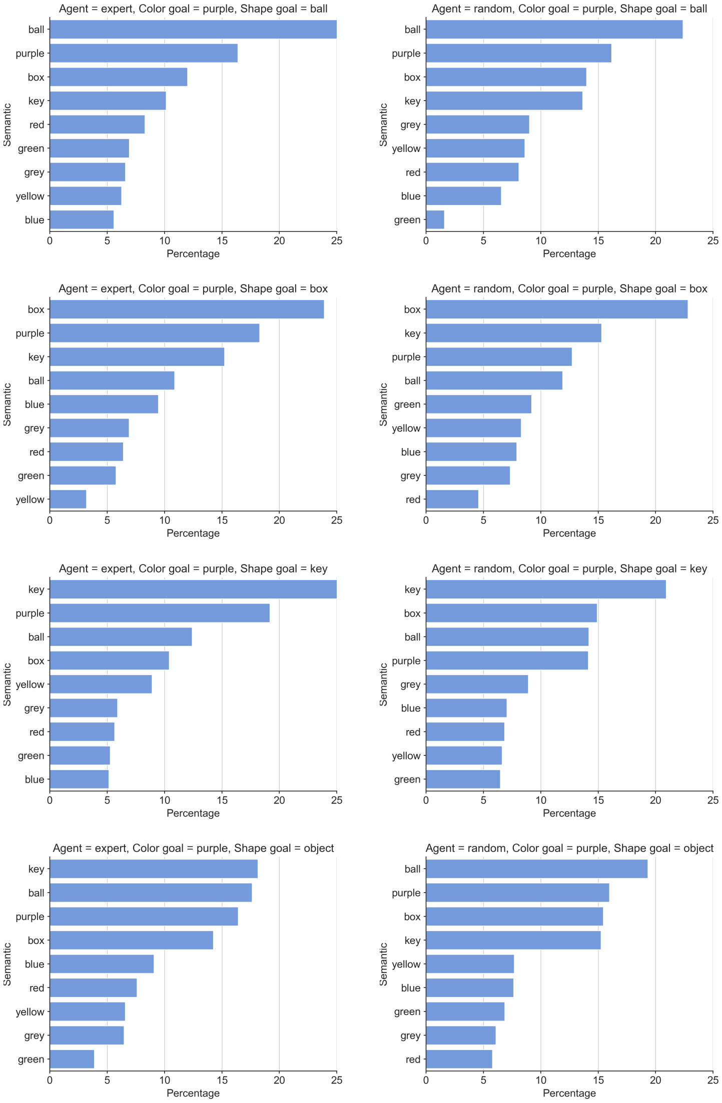
Figure 6: **Left:** Trajectories for the green color goal from BabyAI's built-in expert agent which always reaches the goal. **Right:** Random agent trajectories. In both cases the semantics of the goal are among the most observed semantic features for any given trajectory. This effect is less pronounced in the random agent.



(a) Expert agent

(b) Random agent

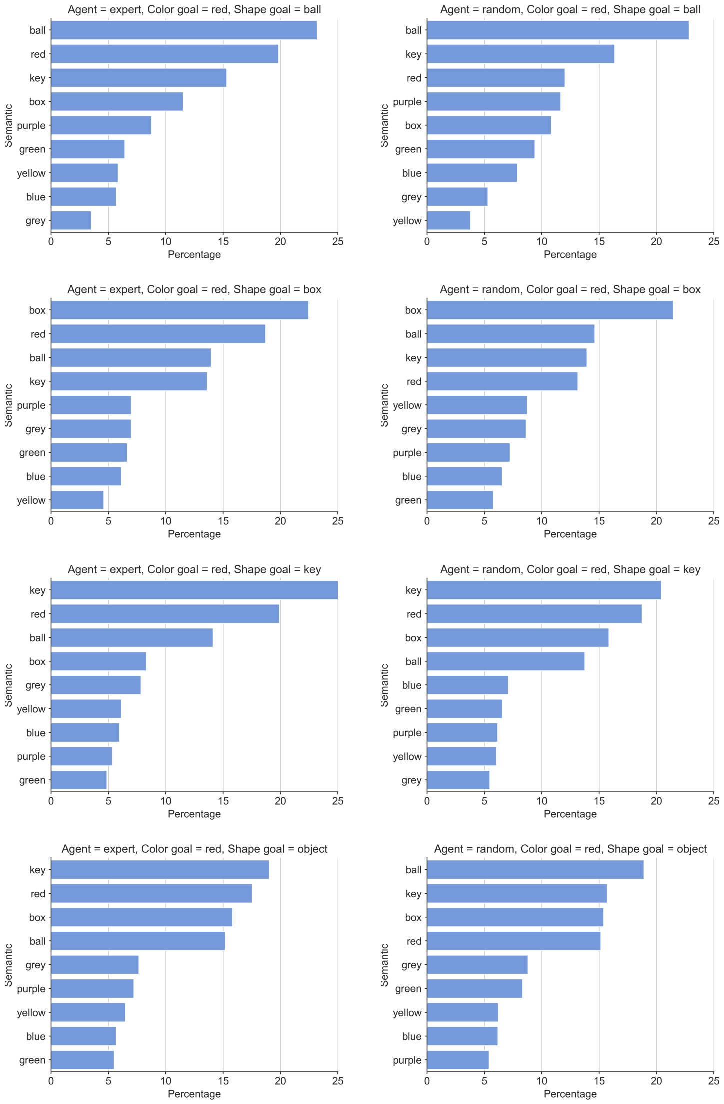
Figure 7: **Left:** Trajectories for the grey color goal from BabyAI's built-in expert agent which always reaches the goal. **Right:** Random agent trajectories. In both cases the semantics of the goal are among the most observed semantic features for any given trajectory. This effect is less pronounced in the random agent.



(a) Expert agent

(b) Random agent

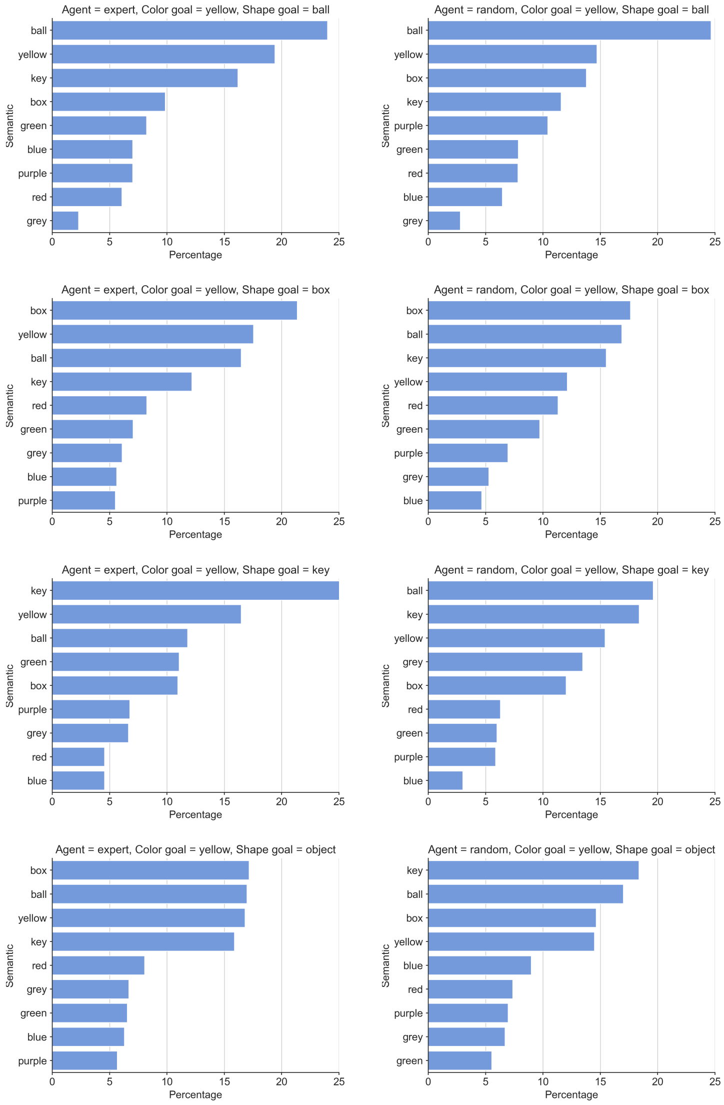
Figure 8: **Left:** Trajectories for the purple color goal from BabyAI's built-in expert agent which always reaches the goal. **Right:** Random agent trajectories. In both cases the semantics of the goal are among the most observed semantic features for any given trajectory. This effect is less pronounced in the random agent.



(a) Expert agent

(b) Random agent

Figure 9: **Left:** Trajectories for the red color goal from BabyAI's built-in expert agent which always reaches the goal. **Right:** Random agent trajectories. In both cases the semantics of the goal are among the most observed semantic features for any given trajectory. This effect is less pronounced in the random agent.



(a) Expert agent

(b) Random agent

Figure 10: **Left:** Trajectories for the yellow color goal from BabyAI's built-in expert agent which always reaches the goal. **Right:** Random agent trajectories. In both cases the semantics of the goal are among the most observed semantic features for any given trajectory. This effect is less pronounced in the random agent.