The Need to Communicate

Divy Bramhecha

4th Year Project Report
Artificial Intelligence
School of Informatics
University of Edinburgh

2023

Abstract

We follow the utilitarian definition of language understanding laid out in, where language is one of the multiple tools available to an agent to accomplish tasks in its environment. Therefore, the success of communication is reduced to the success of the tasks solved by the agents. After developing environments and learning agents for such a paradigm, we show that no artificial means is required to generate structure in the evolved language. Following iterated learning paradigm, we simulate the expressivity of the real environment, along with the pressure to learn. These factors combined we witness the development of compositional language through the utilitarian paradigm. I.e., language learning agents are able to generalize and describe objects never seen by them.

Research Ethics Approval

This project was planned according to the Ethics Policy for Informatics Research Ethics. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Divy Bramhecha*)

Table of Contents

Chapter 1

Introduction

The only way to learn an unknown language is to interact with a native speaker asking questions, holding a conversation that sort of thing [...] If you want to learn the aliens' language, someone [...] will have to talk with an alien. Recordings alone aren't sufficient.

Ted Chiang, Story of Your Life

Language is a distinguishing property of human intelligence, allowing transmission of complex ideas in a variety of contexts. Here we follow the utilitarian definition of language understanding laid out in [11], where language is one of the multiple tools available to an agent to accomplish tasks in its environment. Therefore, the success of communication is reduced to the success of the tasks solved by the agents.

In contrast to this, current natural language tasks often focus on maximizing linguistic objectives on large static datasets, making language an end in itself. A lot of recent advances in Natural Language Processing and Understanding, have been majorly by models maximizing such objectives. However, for agents to interact with, learn from, and help humans in the real world, just capturing such patterns is insufficient [15].

Agents need to communicate successfully to operate together. They also need some kind of language to efficiently collaborate with humans. Moreover, if we developed agents capable of the above, it would also offer us insight into human language evolution and cognition. Similarly, if we are planning to understand this, the agent must learn the language out of the necessity imposed by its environment. This might also help capture the functional aspects of the language, offering an insight in formulation of our existing languages. Moreover, such an environment also makes the evaluation of language straightforward, as opposed to statistical language models, which often require human evaluation.

In this work, we create a set of environments based on Open-AI's physics framework Mutli-agent particle environments. In these environments, following utilitarian paradigm, we develop and test algorithms capable of solving it. We analyze and understand the language created by artificial means and investigate ways to induce structure. We demonstrate that artificial means of inducing structure, like restricting vocab size are quite different from natural language. Therefore, following the iterated Learning framework we show that the structure in language is a completely unintentional and beautiful phenomenon. We successfully simulate the iterated learning along with the pressures of expressivity naturally, without language or structure as goal. We show that, precisely then compositional and functional natural languages emerger.

1.1 The Environment

1.1.1 Paradigm

Following the utilitarian definition of language, we choose an end-to-end learning environment with multiple language agents, each with their own internal goals. Often, to accomplish these internal goals, help from external agents is required. Moreover, these goals are grounded in non-linguistic objectives as discussed in [11], like reaching a particular location. In an ideal scenario, at least one agent would speak a fixed conventional natural language (e.g. English), and other agents are then tasked to solve goals in the environment, along with jointly learning this language. Often it might be possible to include such a fixed language agent without humans, by having fixed hard-coded English, however, such approximations are often crude. Therefore, we believe that involving humans would be an extremely effective way to train such agents.

1.1.2 Related work

A majority of prior work in emergent communication has focused on Lewis signaling and referential games. In such games the listener is tasked with picking an image/object from a set of distractors by listening to a speaker who is tasked to communicate this well. This concept was introduced initially by [24] and recently has been studied in [23], [13].

This setup however suffers from several simplistic assumptions.

1. Listener performs a single action and observes immediate feedback, while in the real world agents must perform long sequences of actions and observe delayed reward.
2. The agents solve a single task, while in the real world agents must perform multiple tasks that partially overlap.
3. The dynamics of both listener and speaker remain static and their is no progression in an episode.

These result in a simple optimization problem, and therefore most of these research utilize standard policy gradient algorithms, like REINFORCE, despite the non-stationarity of agents during training.

[15] showed emergence of communication in an interactive 2d world. However, in this work they assumed a model-based setup, where the world dynamics are known and differentiable. Moreover, learning is conducted through back-propagation through time across the entire episode. This significantly limits the applicability of their work, since knowing differentiable world dynamics of all agents is not always possible.

The same authors try to address this issue in a follow-up work Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments [26]. The MADDPG algorithm proposed here works for a variety of multiagent tasks however, from the list of these environments they completely omit the complex environments mentioned in [15]. This is probably a testament to the complexity of tasks aimed at in [15], and that no general model free reinforcement learning algorithm has been shown to work well in them.

1.1.3 Our environment

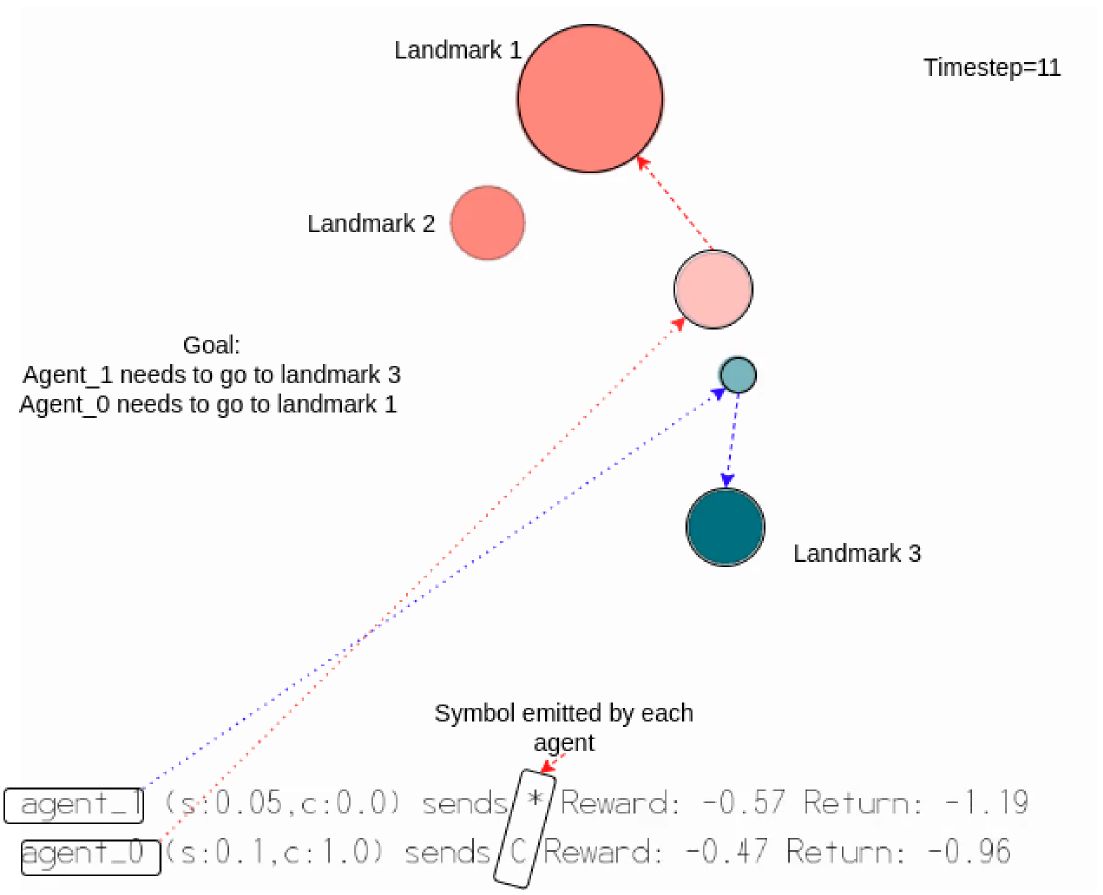
Landmark 1
Timestep=11
Landmark 2
Goal:
Agent_1 needs to go to landmark 3
Agent_0 needs to go to landmark 1
Landmark 3
Symbol emitted by each agent
agent_1 (s:0.05,c:0.0) sends * Reward: -0.57 Return: -1.19
agent_0 (s:0.1,c:1.0) sends C Reward: -0.47 Return: -0.96

Figure 1.1: Environment with 2 agents and 3 landmarks

As part of [26], they released a basic set of environments, along with its source code. This makes it easy to create new environments consisting of new objects and tasks. This code base has further been fixed and maintained by [40] offering an even easier and safer way to utilize these set of 2d environments. Moreover, this allows us to extend the environments to model a more complex set of environments suitable for the task of language evolution. These were specially inspired by environments in [15]. Therefore, using the base code laid out above, we create the following environments:

1.1.3.1 Objects

A 2d environment consisting of Agents, Landmarks and Obstacles. Each of these have the following properties:

1. Circles of radius r.
2. Occupy a position p and are collidable.

3. Have a color c.
4. Agents, also have a velocity described by $\bar{p}$.

Therefore each object x in environment is represented by: $x_i^t = [p, c, r, \bar{p}]$.

1.1.3.2 Agent Actions

The action has a discrete action space, which combines its:

1. Direction of velocity 1 (Up, Down, Left, Right, Nothing).
2. And 1 discrete symbol s from a vocab of size K which is globally visible to all other agents at each time step.

Therefore, the discrete action space of each agent is of size $a_i = 5 * k$

1.1.3.3 Goals

Each agent is assigned an internal goal vector g_i.

Each agent is assigned a color of the landmark they are required to visit by some other agent. This color is visible to all other agents except themselves. Similarly, each agent is tasked to help an agent reach that particular landmark.

Precisely, the goal vector for an agent consists of the color of the landmark and the color of the agent.

This embodies the principle of ”necessity”, as for each agent to accomplish their goal, it requires cooperation from other agents, particularly in form of communication.

Moreover, each agent requires an understanding of both the agents, and the landmarks present, their association with each other and the ability to communicate with each other the correct information.

1.1.3.4 Observation

For each agent a_i, the observation state O consists of:

1. Personal velocity $p = [p_x, p_y]$.
2. Goal vector g_i.
3. All m landmarks $L = [l_1, .., l_m]$ where $l_j = ({}_ix_j, c_j, r_j)$.
4. All n agents $A = [a_1, .., a_n]$ where $a_j = ({}_ix_j, c_j, s_j)$.

Here, s_j is the one hot encoding of the symbol emitted by agent j. Similarly, ${}_ix_j$ is observation of x_j physical state from x_i physical frame, after applying a rotation matrix, to match the perspective. This is to prevent agents emitting coordinates of objectives, without needing to describe them with properties such as color and size.

The final observation for the agent a_i is a concatenation of the above:

$$O_{a_i} = [p_i, g_i, L, A]$$

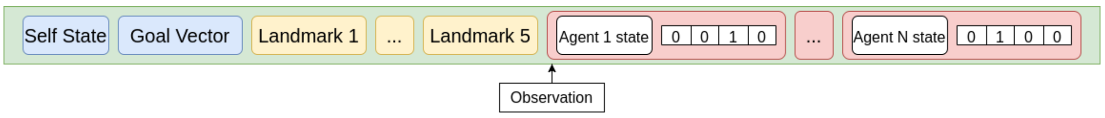

Figure 1.2: Observation Vector for an Agent

1.1.3.5 Reward

Each agent is rewarded for achieving it's own objective of helping the other agent reach its landmark. They are also penalized for uttering a symbol at a particular time step, to simulate effort. All agents are collectively rewarded for the success of all other agents.

Formally, at each time step, t reward is the negative distance of each agent from the center of the landmark. If the agent is within the radius of the landmark, it is given a reward of 0. Furthermore,t if the agent utters a symbol other than silence, it is penalized by -0.02.

We choose nonsparse rewards to make the optimization problem simpler, as the main focus of our work is on the non-stationarity and communication in a multiagent setting.

1.2 Task

We wish to be able to create algorithms capable of developing agents which can communicate and collaborate with any type of agents (including humans) to achieve/ solve objectives of complex environments. To do this the agents should be capable of:

1. *Developing communication* — Develop ways to communicate given the tools of the environment.
2. *Learning communication* — Learn the ways of communication of already existing agents.
3. *Emerging syntactical strucutre through evolution accross generations* — Investigating Iterated learning without biases.

Developing communication amongst agents have been explored in various research projects; however, many of these suffer from simplistic environment assumptions mentioned in 1.1.2. Moreover, works which get rid of these environmental assumptions don't emphasize on the emerging languages and often demonstrate success of communication through rewards achieved by the agents in the environment. [] show that simply reward is not often sufficient to show emerging languages, or even successful communication. The research work by Igor in [15] shows how these hurdles can be overcome; however, he assumed a model-based reinforcement learning algorithm, heavily limiting the model's ability to be applied to different environments. These limitations are discussed in **??**. Despite the algorithm not being generally applicable to solving the main task, it's results can be refered to when developing model-free algorithms. They showed the emergence of basic compositional language in these model-based agents. The language was represented as a stream of abstract discrete symbols over time, demonstrating properties of structure, vocabulary and syntax. They showed emergence of these properties despite the lack of any explicit language goals or predesigned meaning associated with any of the symbols, nor any explicit roles as speaker/listener models, nor any explicit turns to structure any dialog.

Creating an algorithm capable of above without tying it down to a particular environment would bring us significantly closer to achieving the task of *Developing communication*, and a major part of this work will focus on that.

The importance of *Learning communication*, is that it allows us to develop agents to learn human languages, too, by treating them as just another learning algorithm. The advantage of being able to learn communication recognizes the concept of allowing agents to have private goals, yet requiring collaboration to achieve them. If the goals of the agents are shared and known, then in an ideal solution there would be no need for communication. As described in 1.1.1, an ideal way to train each agent for communication with humans would be to include one of the humans as agents. However, at the current state of research all reinforcement learning methods are too sample inefficient for this to feasible. Similarly, creating rule-based natural language agents would lead to various biases, worsening the performance. Moreover, creating good rule-based agents would take significant amount of time, and it's results would not scale up to the goal of creating grounded language-based agents.

Therefore, before humans can be involved in the training of such agents, it's important to ensure that our algorithms are sample efficient enough. Therefore, we tackle this by making the agents learn to communicate with a group of agents who have already developed their own set of languages to solve the task. More importantly, measuring how fast can an agent who has already developed and learnt a particular language, can adapt in a group of agents with different language of varying complexities.

Chapter 2

Background

2.1 Single Agent Reinforcement Learning

RL is the interaction of an agent with an environment using Markov decision processes (MDPs). It can be described as a tuple of:

$$(S:\text{states}, A:\text{action space}, R:\text{reward-func}, T:\text{transition-func})$$

The transition function $T : S \times A \times S \rightarrow [0,1]$ is the probability of a transition from any state $s \rightarrow s'$ where $s, s' \in S$ given $a \in A$.

The reward function $R : S \times A \times S \rightarrow R$ defines the immediate and possibly stochastic reward that each agent would receive given the actions a each agent has executed in state s and the state s' it transitioned to.

MDPs are sufficient models to obtain optimal decisions in a single agent fully observable environment. Solving MDP gives us a policy $\Pi : S \rightarrow A$. An optimal policy Π^* maximizes the expected discounted sum of rewards, i.e. the returns.

2.1.1 Q-Learning

Q-learning was developed for stationary single agent, fully observable environments with discrete actions. A Q-learning agent stores the estimate of its returns starting in state s, taking action a as $Q(s,a)$. Q^* is an optimal function that maps state-action (s,a) pairs to the returns when the agent is following an optimal policy. This is estimated with a tabular entry $Q(s,a)$.

For each state transition $(s,a) \rightarrow s'$ and reward r the Q-table is updated as follows with the learning rate $\alpha \in [0,1]$.

$$\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha[r + \gamma \max \hat{Q}(s',a') - \hat{Q}(s,a))]$$

Q-learning is proven to converge to Q^* if state and action spaces are discrete and finite and the sum of the learning rates goes to infinity.

2.1.2 REINFORCE

REINFORCE in contrast is a policy gradient method, which learns to parameterize policies directly without using intermediate value estimates [39]. In policy gradient methods, policy parameters are learned by following the gradient of a certain performance measure with gradient descent.

Reinforce uses estimated return by Monte Carlo (MC) methods with full episode trajectories to learn policy parameters θ, with $\Pi(a;s,\theta) \approx \Pi(a;s)$, as follows where G_t represents the return, α is the learning rate, and $A_t \approx \Pi$.

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t;S_t,\theta_t)}{(A_t;S_t,\theta_t)}$$

Policy gradient methods are shown to have high variance [20].

2.1.3 Actor Critic

We can generalize the policy gradient by including a comparison to an arbitrary baseline of the state. This baseline can be any function invariant to the action. A natural choice for the baseline function is a learned state-value function. Moreover, this could also help lower the variance. Furthermore, when using a state-value function for bootstrapping, it assigns credit and critizies the action selected by the policy.

Such a family of methods, where the actor is the policy and a critic is the value function, are known as actor-critic methods. An example of an actor-critic algorithm is the Deterministic Policy Gradient. In DPG, the critic is similar to Q-learning, and the actor is updated based on the gradient of the policy's performance.

2.1.4 Deep-Q Learning

The above methods promise guaranteed convergence to optimal policies. However, these convergence promises are for infinite steps and are often too slow to learn in large state spaces. This is because the methods do not generalize across the state space. Moreover, often in practice, state representations need to be hand-specified [39].

These challenges can be addressed by using deep learning as function approximators. For instace, $Q(s,a;\theta)$ can be used to approximate the value function, with θ as the parameters of our neural network [3].

This helps tackle the above problems, since the neural network can both generalize across state space and learn to represent the state features.

However, extending deep learning to RL creates additional challenges, particularly non-i.i.d. data, since many supervised learning methods assume that training data follow an i.i.d. stationary distribution [2].

In RL since data is generated through sequential interactions with environment it tends to be highly correlated violating the independence condition. In addition, as the agent actively learns while exploring different parts of the state space, the sampled data are also nonstationary, violating the condition of the sampled data being identically distributed [29].

In [10], the authors showed that non-linear function approximators poorly estimate the value function. They showed problems like over/underestimation, instability, and even divergence of the function approximation due to the delusional bias that ”a backed-up value estimate is derived from action choices that are not realizable in the underlying policy class.”

Deep Q-Network (DQN) tried to alleviate some of these issues, successfully demonstrating the power of function approximation in RL. It used a deep neural network for function approximation and maintained an experience replay (ER) buffer to store interactions (s,a,r,s'). It also used an off-policy method to update its model parameters θ by maintaining an additional copy θ'. This is to help stabilize the learning, by reducing the non-stationarity of the data distribution. At each training step, DQN minimizes the mean-squared error between the Q-network and its target network using the loss function:

$$L_i(\theta_i) = E_{s,a,r,s'}[(r+\gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i))^2]$$

where target network parameters θ^- are set to Q-network parameters θ periodically and mini batches of (s,a,r,s') tuples are sampled from the ER buffer.

2.1.5 Deep Deterministic Policy Gradient

For physical control tasks, the action space is often continuous. In such tasks DQN is not suitable. Deep Deterministic Policy Gradient (DDPG) is a model-free off-policy actor-critic, based on the DPG algorithm described above. Additionally, it proposed a new method for updating the networks, where the target network parameters are updated gradually at all times, in contrast to the hard update (direct weight copy) used in DQN. Since DDPG is off policy, the exploratory behavior is controlled by adding sampled noise to the actor’s policy.

2.1.6 TRPO and PPO

TRPO [33] is a class of policy gradient methods that restricts the update of policies to within the trust region of the behavior policy by enforcing a KL divergence constraint on policy update at each iteration. Formally, TRPO restricts the update to:

$$\text{subject to:} E_{s_t,u_t}[KL(\pi_{\theta_{\text{old}}}, \pi_\theta)] \leq \delta$$

here, θ_{old} are the policy parameters before the update. This forumulation hoever is computationally expensive as it requires multiple Hessian-vector products for nonlinear conjugate gradients when approximating the KL constraint.

Therefore, PPO [35] approximates this constraint using clipped policy ratios. The policy loss now becomes:

$$L = E_{s_t,a_t}[\min(\frac{\pi_\theta(\mu_t|s_t))}{\pi_{\theta old}(\mu_t|s_t))}A(s_t,a_t), clip(\frac{\pi_\theta(\mu_t|s_t))}{\pi_{\theta old}(\mu_t|s_t))}, 1-\varepsilon, 1+\varepsilon)A(s_t,a_t)]$$

They also introduce a variant to the advantage function called Generalized Advantage Estimation. Here,

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + ... + ... + (\gamma\lambda)^{T-t+1}\lambda_{T-1}$$

$$\delta_t = r_t + \gamma V(s_{t+1} - V(s_t))$$

. They claim that this advantage function reduces variance and helps with RNN policies better [35].

Another advantage of PPO is that it can be used in a distributed fashion, i.e, Distributed PPO (DPPO). Such parallelization enables utilizing multiple cpu cores for efficient training data generation. Techiniques, used in distributed approaches like A3C can be applied to here for faster processing.

2.2 Multi-Agent Reinforcement learning

Learning in a multi-agent environment is naturally more complex than in single-agent case, as agents have to interact with the environment and each other.

Extending single agent RL algorithms, directly to a multi-agent setting is reffered to as the independent learners aproach, a.k.a. decentralised learners approach. These violate multiple assumptions made in the single agent setting. Particularly, the Markov property becomes invalid – the future dynamics, transitions, and reward no longer depend only on the current state – since the environment is no longer stationary [22]. It can fail entirely when another agent learns and adapts based on previous interactions. Despite the lack of guarantees, independent learners have been used in practice, providing advantages with respect to scalability while often achieving good results [28].

To understand why multi-agent domains are non-stationary from an agents' local perspectives, consider a Markov game (S, N , A, T , R), which can be seen as an extension of an MDP to multiple agents [25]. An important distinction is that the transition function T and the reward function r depend on the actions $A = A_1 \times A_2 \times .. \times A_N$ of all N Agents. For a learning agent i the value function similarly depends on the joint action and the joint policy, and consequently the optimal policy depends on the policy of all other agents. Specifically, the other agent's policy can be non-stationary, i.e. changes as the agent learns.

Multi-agent scenarios also give rise to commonly known problems such as action shadowing [42], the curse of dimensionality, and multi-agent credit assignment [1].

2.2.1 MADDPG

Lowe et al. [26] noted that using standard policy gradient methods on multiagent environments yield high variance and performs poorly. The variance experienced is further increased by the presence of all the other agents. This is because all the rewards of the agents depend on the rest of the agents, and they formally show that as the number of agents increases, the probability of taking a correct gradient direction decreases exponentially [26].

To overcome this issue, they proposed the Multi-Agent Deep Deterministic Policy Gradient (MADDPG), which builds on top of DDPG, to train a centralized critic per agent that receives all agents' policies during training to reduce variance by eliminating the non-stationarity caused by concurrently learning agents.

The actor only has local information (turning the method into a centralized training with decentralized execution) and the ER buffer records experiences of all agents. MADDPG was tested in both cooperative and competitive scenarios, and the experimental results showed that it performs better than several decentralized methods (such as DQN, DDPG, and TRPO). The authors mention that traditional RL methods do not produce consistent gradient signals. This is exemplified in challenging scenarios where agents continuously adapt to each other, causing the learned best-response policies to oscillate.

2.2.2 COMA

Another approach based on policy gradients is the Counterfactual Multi-Agent Policy Gradients(COMA) [8]. Here, they compute a counterfactual baseline to marginalize the action of the an agent while keeping the actions of the agents fixed. Then, an advantage function can be computed comparing the current Q value to the counterfactual. This counterfactual baseline is similar to difference rewards [41] which is a method assigning credit to each agent in a cooperative multi-agent team. They particularly use aristocratic utility, which aims to measure the difference between the actual action of an agent and the average action [43]. These methods help address both the non-stationarity problem and the multi-agent credit assignment problem.

2.2.3 QMIX

The above-mentioned multiagent approaches suffer from scalability, as opposed to independent learners. There is a hybrid between the two approaches that learn a centralized but factored Q value function [12]. Value Decomposition Networks (VDNs) [38] decompose a team value function into an additive decomposition of the individual value functions. QMIX [32] builds on the above idea, however, instead of sum, it uses a mixing network that combines the local values in a nonlinear way, which can represent monotonic action value functions.

While the mentioned approaches have obtained good empirical results, it is still an ongoing research topic in MARL.

2.3 Learning Communication

There have been various attempts to communicate in cooperative settings. A majority of these make assumptions not possible in the real world, such as: sharing of network parameters [9] or access to hidden states of other agents [36] [16], or predesigned communication architectures which can restrict emergent communication [31]. Moreover, enforcing a specific communication architecture can limit the diversity of emergent communication protocols [21].

Below we discuss important works in communication as they offer us insight and ideas to better improve these exisiing models.

2.3.1 DIAL

DIAL [7] was the first to propose a learnable communication through backpropagation with deep Q networks. At each timestep, an agent generates its message as the input of other agents for the next timestep. Gradients flow from one agent to another through the communication channel, bringing rich feedback to train an effective channel. However, the communication of DIAL is rather simple, just selecting predefined messages.

Further, communication in terms of sequences of discrete symbols are investigated in [14] and [15]. [15] uses model based reinforcement lerning, assuming that the world dynamics are known and differentiable. They then train the models by backpropogating through the entire episode, including world dynamics. However, this is not possible for most scenarios. For [14] their environment/ game is a single-step game offering a simple optimization problem. Therefore, methods like REINFORCE are sufficient to solve their environment, not offering further insight if the methods scale to more complex environments.

2.3.2 CommNet

CommNet [37] is a large dense network that maps input from all agents to their actions, where each agent occupies a subset of units and can broadcast to a communication channel to share information. At a single communication step, each agent sends its hidden state as the communication message to the channel. The averaged message from other agents is the input of next layer.

2.3.3 BiCNet

BiCNet [31] is an actor-critic based model for continuous actions. It uses rnn's to connect each individual agent's policy and value networks. BiCNet is able to handle real-time strategy games such as StarCraft micromanagement tasks. However, it assumes that the agents know the global states of the environment, which is unrealistic in practice. Moreover, predefined communication architectures restrict communication and hence restrain potential cooperation among agents.

Chapter 3

Body

3.1 Core implementation

Previously mentioned centralized multiagent approaches like COMA do not suffer from non-stationarity, however, have scalability issues. However, independent learning agents which are better suited to scale suffer from non-stationarity issues.

However, the extent of the non-stationarity seems unclear, and IPPO [5] and MAPPO [44] simultatneously show that the independent learning algorithm PPO, and it's code-level optimizations greatly alleviate this issue. Moreover, they emperically show that both IPPO and MAPPO outperform state-of-the-art multiagent algorithms on a variety of tasks. These include the basic environments proposed in [26]. Each of the algorithms is based on PPO described in 2.1.6. MAPPO, shares the learning and parameters of all their agents. Although this aids learning, and the sample efficiancy of the overall agents, it is not possible when interacting with different agents. Furthermore, [27] show that the popular centralized critic design in the current literature for MARL is not strictly beneficial.

Therefore, here we use PPO to learn decentralized policies π^a for agents with individual policy clipping as described in 2.1.6.

We use a variant of the independent learning advantage function, where each agent a learns a local observation-based critic $V_\phi(z_a^t)$ parameterised by σ using Generalized Advantage Estimation (GAE).

In the following, we highlight all the code-level optimizations needed to enable the core algorithm to perform well. These seemingly insignificant algorithmic changes to the core policy gradient method turn out to be critical, without which the algorithms are found to never solve the tasks. [44], [5], [6] carry out various ablation studies demonstrating the importance of techniques presented below. Similarly, we also confirm that the hyperparameter choices found in them work best for us as well.

3.1.0.1 PPO Clipping

A major trick in PPO is the use of clipped importance ratio and value losses. It is an attempt to constrain the policy and value functions from drastically changing between iterations, a weak approximation of the TRPO. The strength of the clipping is controlled by a hyperparameter. [44] Hypothesizes that policy and value clipping, along with the number of training epochs, controls the nonstationarity caused by changing multi-agent policies. Later, they showed that lower values slow the learning speed, and they correspond to a more consistent policy improvement. However, higher values result in greater variance and greater volatility in performance.

3.1.0.2 Value function clipping

Additionally, the values are also clipped to restrict the update of the critic function for each agent a to be smaller than ε as proposed by [34]

$$L_v = min[(V_{\theta t} - V_{target})^2, clip(V_{thetat}, V_{thetat-1} - \varepsilon, V_{thetat-1} + \varepsilon) - V_{target})^2)\}]$$

The update equation restricts the update of the value function within the trust region, and therefore helps us to avoid overfitting to the most recent batch of data.

3.1.0.3 Reward scaling

Rather than feeding the rewards directly from the environment into the objective, we perform a discount-based scaling. Here, the rewards are divided through by the standard deviation of a rolling discounted sum of the rewards, i.e. without subtracting and re-adding the mean.

3.1.0.4 Value Normalization

To stabilize learning [44], suggests we standardize the targets of the value function by using running estimates of the average and standard deviation of the target values. This stabilizes the value targets, which, without normalization, can drastically change during training. Concretely, during value learning, the value network will regress on the normalized target values. When computing the GAE, we will use the running average to denormalize the output of the network so that the value outputs are properly scaled. They find that using value normalization never hurts training and often significantly improves the final performance.

3.1.0.5 Orthogonal initialization and layer scaling

[6] suggests that instead of using the default weight initialization scheme for the policy and value networks, the implementation uses an orthogonal initialization scheme with scaling that varies from layer to layer. This significantly helpful with ReLU based activation functions.

3.1.0.6 Adam learning rate annealing

The implementation anneals the learning rate of Adam. This both helps smoother convergence, but also helps tackle catastrophic forgetting. This can happen sometimes when the KL-divergence get's too large.

3.1.0.7 Global Gradient Clipping

[6] suggests that after computing the gradient with respect to the policy and the value networks, they clip the gradients such that the ”global l2 norm” does not exceed 0.5.

3.1.0.8 Training Data Usage

A major trick in PPO is the use of importance sampling to perform off-policy corrections, allowing for sample reuse. After a batch of samples is collected, [35] suggest splitting the large batch into mini-batches and training for multiple epochs. In continuous control domains, the common practice is to train for tens of epochs with around 64 mini-batches per epoch. However, the [44] found that this significantly degrades the performance when the samples are used too often in multiagent setting. It could be due to the non-stationarity issue in MARL.

Additionally, using more data to estimate gradients leads to improved practical performance. Therefore, we do not split the batch into mini-batches.

3.1.0.9 Overall Loss Includes Entropy Loss

The overall loss is then calculated as

$$L = L^a(\theta) + \lambda_{critic} V + \lambda_{entropy} * H(\pi^a)$$

which includes entropy of policy $H(\pi^a)$ maximization, which intuitively encourages exploration by encouraging the action probability of the distribution to be more chaotic.

3.1.1 Techniques specific to enable langage learning.

The major, and arguably difficult part of our enivornment task is the agent's ability to communicate with other agents. Therefore it is essential that the agents are able to understand the cues to optimize towards them. Not only is our environment non-stationary, but the sequential communication nature also violates the Markovian property. Moreover, as the number of agents and the properties of landmarks increase, these problems are significantly exemplified.

3.1.1.1 RNN based network across timesteps to better capture communication.

Our models only take in a single frame at each timestep. Since it is beneficial to not communicate all time, they need a way to remember the last communication they received from the other agents.

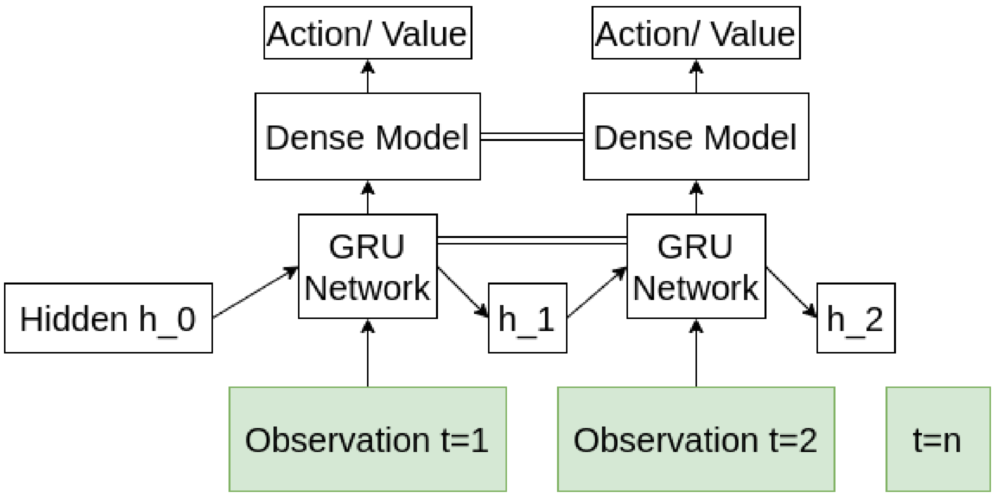

Figure 3.1: Recurrent PPO.

Moreover, initially these messages will be random and noisy, therefore they need to act upon these sequential noisy messages to find an underlying pattern. If they only act and learn from the current message, it will be infeasable to optimize independently. Moreover, for PPO which does not have a large replay buffer, the non-stationarity of the problem can take over.

Recurrency acts as an explicit mechanism to do just that. Therefore, we extend the PPO model proposed by [35] and described above to include a recurrent policy.

Here too, the implementation of the Recurrent Policy played a large difference. Following the description in Open-AI Baselines, they suggest to treat the hidden state analogous to an additional observation, and to store it as part of importance sampling. Following this policy is updated, as usual, using the stored hidden states for each timesteps. However, since we are updating over multiple epochs, this leads to stale hidden states. For the task of communication, this likely leads to much greater variance, causing the model to fail.

Therefore, next we implement the RNN policy instead to treat the hidden state as a latent variable which has to to be recomputed for every new episode/ computation. This means the hidden state is no longer stored with importance sampling, and during the policy update, a new hidden state is initialized for each starting observation. Following which, the loss functions are additionally sum over time, and the networks are trained via Backpropagation Through Time (BPTT). The model is illustrated in figure 3.1.

3.1.1.2 Prediciting future action and communication of other agents

The recurrent model allows our agents to distinguish the noise in the early stages of emergent communication.

We hypothesize that to further reduce the expected variance, we assign the model to an auxiliary for predicting the future action and the communication symbols of other agents. Doing so, helps the model solve an additional relevant optimization task, better reusing the data and speeding up the training process.

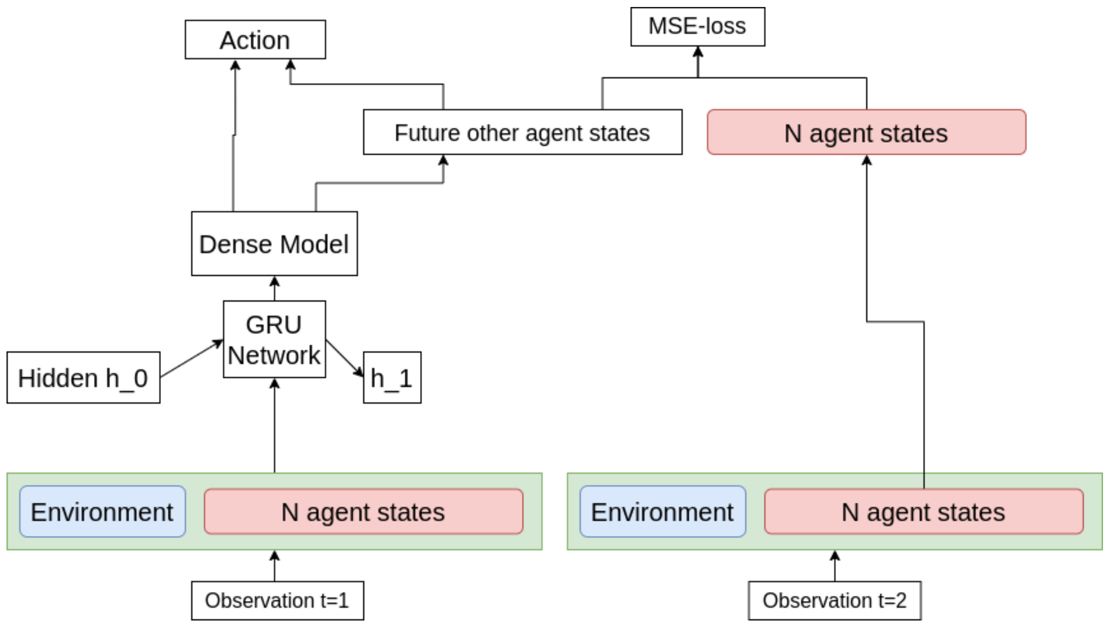

Figure 3.2: Future loss

Moreover, it could allow the model to perform informed exploration as described in [30].

This is illustrated in figure 3.2.

3.1.1.3 Bidirectional rnn based attention mechanism

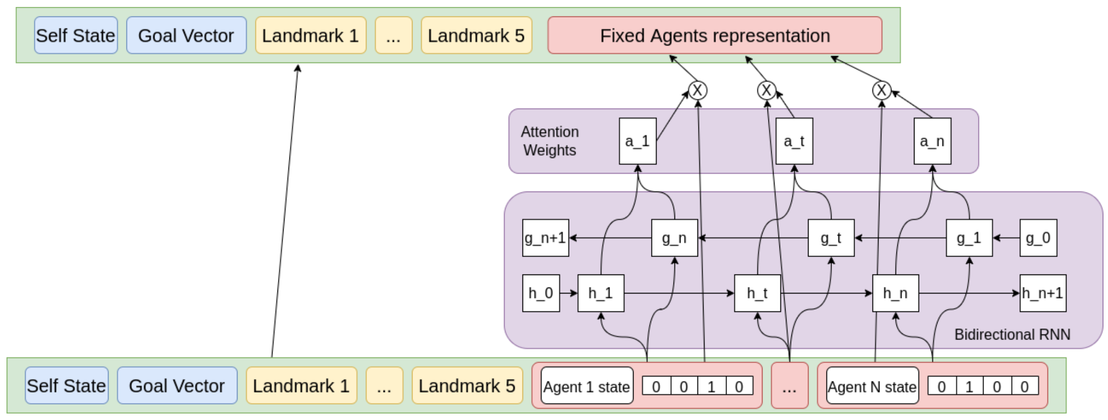

Figure 3.3: Capturing attention over Agent states. The observations are the same as 1.2

We addressed the issue of noise in sequential communication by recurrency through time. However, as we scale up the number of agents from 2 to 4, the noise in the communication increases significantly, hampering any learning.

Not only do non-stationarity and credit assignments become increasingly challenging, the agents are now additionally required to decipher which agent's communication to pay attention to. This slows down the learning exponentially with respect to the number of agents.

Moreover, the observation space also increases significantly, needing to account for each agent's one hot-encoded communication vector.

Therefore, we use a bidirectional rnn over various communication channels. The mean of output of the bidirectional rnn over each communication channel is taken as the attention over each communication channel. The attention weighted mean forms a fixed size representation. This is illustrated in figure 3.3.

Most notably, this allows agents to interact in environments with a varying/ interchangeable number of agents, without requiring retraining from scratch.

3.1.2 Experiments and results

We use MADDPG and QMIX as our baseline comparisons for all our experiments. The best hyperparameters for these were picked as described in [44].

3.1.2.1 Hyperparameter Selection

For our model, we perform a sweep of hyper parameters, both to verify the findings of [44], [5], [6] and to pick a stable set of hyper parameters for all our future experiments. To find these, we perform a sweep of parameter search as described in 3.1

Hyperparameter Name	Hyperparameter Sweep values	Selected Value
Environment	{2-Agent, 3-Agent, 4-Agent}	
Learning Rate	{1e-3, 7e-4, 4e-4, 1e-4}	7e-4
Training Epoch	{5, 10, 15}	10
Activation	{ReLU, TanH, SELU}	ReLU
Clip	{0.05, 0.1, 0.2}	0.1
Entropy coef	0.1	0.1
Network	mlp, gru	gru
Gru hidden dimension	64, 128	128
num envs	64, 256, 512, 1024	512
FC layer dim	256	256
FC layer count	2, 3, 4	4
Attention	None, mlp, bidirectional-rnn	bidirectional-rnn
Future State Loss	True, False	True

Table 3.1: Sweep performed for hyperparameter search for decentralised PPO.

Note, the best hyper-parameter values we found are very similar to those found in [44].

3.1.2.2 Ablation Studies

We empirically justify the design decisions for our language learning agent. Particularly, our choice of recurrency, future agent action, prediciton, and bidirection-rnn based attention for multiple agents.

The models were first trained on 2 agent scenarios for 1 million timesteps. Next, the models were trained on 3 agent scenarios for 1.5 million timesteps. Lastly, the models

2 Agent Scenario	Episode Return	End Reward	Symbols/ Episode
MADDPG	-9.1	-0.07	8.1
PPO	-15.3	-0.46	23.2
PPO + GRU	-8.4	-0.05	2.1
PPO + GRU + Future Agent State	-7.1	-0.00	1.2
PPO + GRU + Future Agent State + Attention	-9.2	-0.08	7.6
3 Agent Scenario	Episode Return	End Reward	Symbols/ Episode
MADDPG	-10.6	-0.12	12.6
PPO	-15.7	-0.51	19.1
PPO + GRU	-9.6	-0.08	6.8
PPO + GRU + Future Agent State	-8.2	-0.05	4.5
PPO + GRU + Future Agent State + Attention	-9.1	-0.07	7.9
4 Agent Scenario	Episode Return	End Reward	Symbols/ Episode
MADDPG	-12.1	-0.35	16.7
PPO	-15.4	-0.53	19.5
PPO + GRU	-13.8	-0.42	21.4
PPO + GRU + Future Agent State	-11.9	-0.29	20.6
PPO + GRU + Future Agent State + Attention	-9.1	-0.08	12.6

Table 3.2: Ablation study for architecture choice. Results are the mean of 3 different seeds. Note: the standard deviation for return values is ≤ 0.6, for end reward ≤ 0.02 and for Symbols/Episode ≤ 2.3

were trained on 4 agent scenarios for 2 million timesteps. Note: For the Bi-directional attention model, since the model is invariant in structure to the number of agents, the weights were transfered from 2-agent scenario to 3-agent scenario, to the 4-agent scenario.

Note on reward values: The end-reward is reward an agent receives at the end of the episode. An end reward of 0 indicates that both agents have successfully communicated and reached their goal destination. Reward ≈ -0.45 indicates that the agents are unable to communicate well and have settled at the midpoint of landmarks. And a Reward ≤ -0.6 indicates the agents are failing completely.

We see in Table 3.2, that decentralized PPO without recurency is always unable to develop communication succesfully. This is likely due to the extremely noisy signals and the lack of a proper way to assign credit. Therefore, the best strategy for it is to go to the center of landmarks.

Once we add a recurrency in form of a GRU network, the model solves the task properly. It already performs better than MADDPG. It has learnt to minimize the number of symbols used and develops a holistic language. This is discussed further in 3.2. The strategy developed by the agents involves moving towards the center of the landmark, while receiving symbols from the agents, once ready to move towards the respective landmark. However, when the future agent state task is enabled, the agent is able to create another much more compact language. As a result, it only needs to emit 1.2 symbols on average throughout the episode, as seen in 3.2. The same emperical benefits are reflected through all environments. We believe that this auxiliary task enables the model to represent the weights in a manner to amplify the goal of language learning. For instance, if an agent emits a symbol and can predict the behaviour of the other agent with respect to that symbol, then it can better explore emitting different symbols. This in turn speeds up the process of convergence.

Lastly, we see that all models fail to solve the task in the 4 agent scenarios. Note: despite the main objective remaining the same, increasing the number of agents offers additional challenges in terms of communication. These include a new necessity to address a listener. Similarly, it offers a challenge of attending to the correct speaker. Interestingly, all models are able to perform well in the 3-agent scenario but fail completely in the 4 agent scenario. Perhaps, the number of steps to learn with increasing number of agents is exponentially high. Therefore, we previously proposed the bidirectional attention mechanism over agents. We see empirically, the model, solves the task of 4 agents successfully. Note, we initialized all 4 agent weights from the 3 agent scenario. This might make the comparions unfair to other agents, however, the precise decision to choose this architecture was based on its ability to transfer weight to environments with any number of agents. Having additional weights, we see the model underperforming slightly for 2 and 3 agent scenarios but continues to retain its performance for the 4 agent scenario, greatly outperforming all other models.

We believe such scalable architectures are essential to grow models into more complex environments through curriculum learning.

3.2 Emergence of structure in language

We set out with the goal to understand language and its evolution. We believe that language exists as a by-product of our need to collaborate. Therefore, we followed the utilitarian definition of language and created an environment suitable to it. We also developed learning algorithms suitable to solve the required tasks in this environment.

The agents have clearly developed some form communication, as evidenced by them succesfully solving the task in the environment. In this section, we further investigate this communication.

3.2.1 Vocab and symbol emission through time.

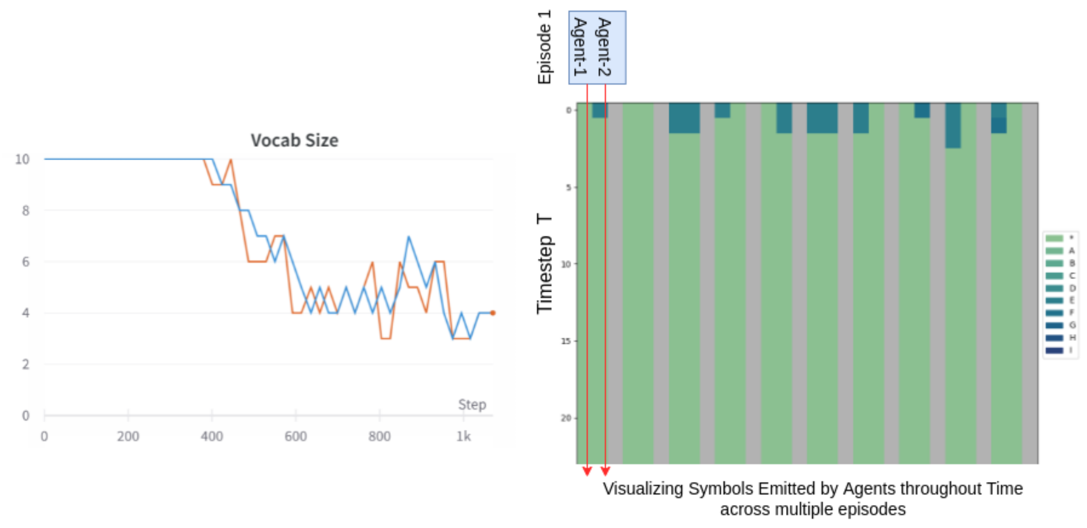

Figure 3.4: 2 agents converging. Left: Vocab size with respect to training timestep, Right: Symbols emmited through time accross 10 different episodes.

Throughtout the experiments, we keep track of various metrics. These include the number of unique tokens used, a.k.a vocab size. We also keep track of the exact symbol emitted by the agent, and at what time step. This graph can be used to understand, and find any simple underlying patterns about the agent's communication.

In Figure 3.4 we observe the vocab size vs training graph for a pair of agents with 3 landmarks with 3 colors. We see that over the duration of training, the vocab size starts to drop slowly after 400,000 timesteps, from 10 to 4. Its at 400,000 steps that the model has become capable of maximizing the reward. Despite, having no objective of reducing vocab size, we see a reduction in it. It is possible that to make fewer mistakes, the agents are settling on set symbols for communicating a particular meaning. Ideally, since there are 3 colors, the number of concepts required would be 3. Upon investigating the videos of the model, we find that the model has developed a holistic language, where each symbol has an exact meaning associated. Its the opposite of compositional. Moreover, we also find that some symbols are associated with the same meaning, like synonyms.

On the right in Figure 3.4 we see the converged plot of few randomly sampled episodes. We see that the models each just utter a symbol or two, indicating the color of the landmark.

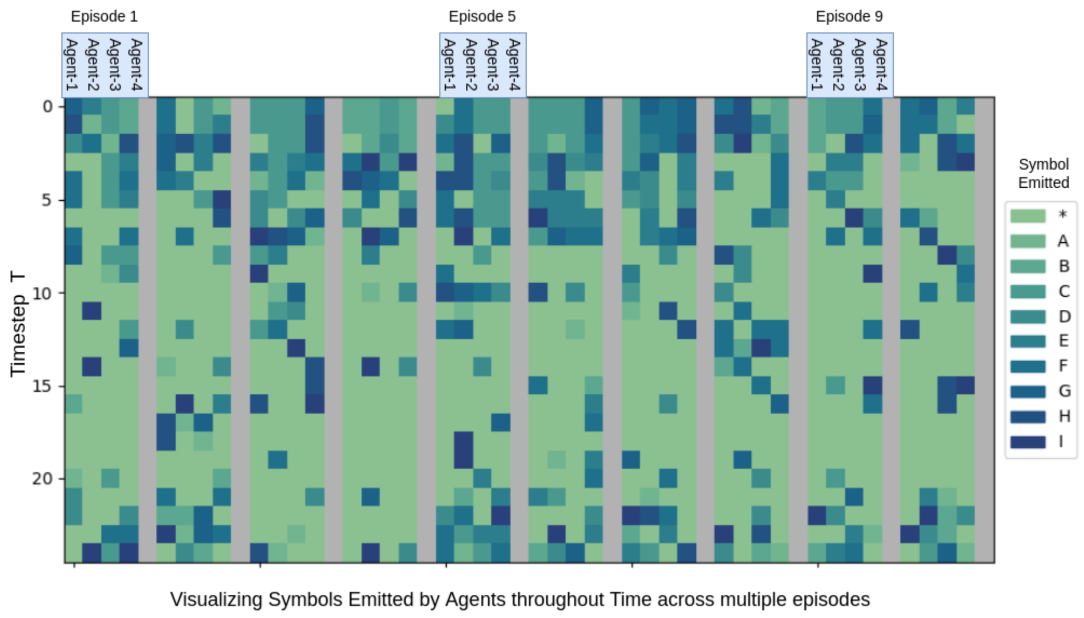

Figure 3.5: Symbols emitted by 4 Agents

In figure 3.5, we see the Attention model converging. We observe that this plot is a lot more chaotic, than 3.4. This is because despite the model running 3 times the number of timsteps than the 2 agent scenario, it still has not converged. Interestingly, we can observe some natural properties. Agents tend to communicate a lot in the beginning to indicate the landmark, guide a little in between, and reinforce the belief of the respective agent towards the end.

3.2.2 Contextual Embeddings

So far, we looked at the videos of the agents to attempt to decipher the meaning of the symbols. However, often due to inconsities accross some examples, this can get tedious. Another way to understand what the model thinks about certain concepts can be visualized by getting contextualized embeddings. Note, these are the embedding representations prior to the action layers.

We generate this by passing artifical vectors to the model. Example for a random observation vector, inputing symbols, landmark colors, or goal vector. Most interestingly, unlike NLP models which are based on only text data, we can get contextualized embeddings in the environment.

On these embeddings, we perform t-sne to find and better visualize the clusters and patterns in the data.

In Figure **??** we see TSNE plot of an agent from 3.4. Here we can see that to navigate to 4 different colored landmarks, the agent relies on the following 4 clusters. Being a continuously interactive environment, it is likely that the context of the observation highly impacts the meaning of the symbols too.

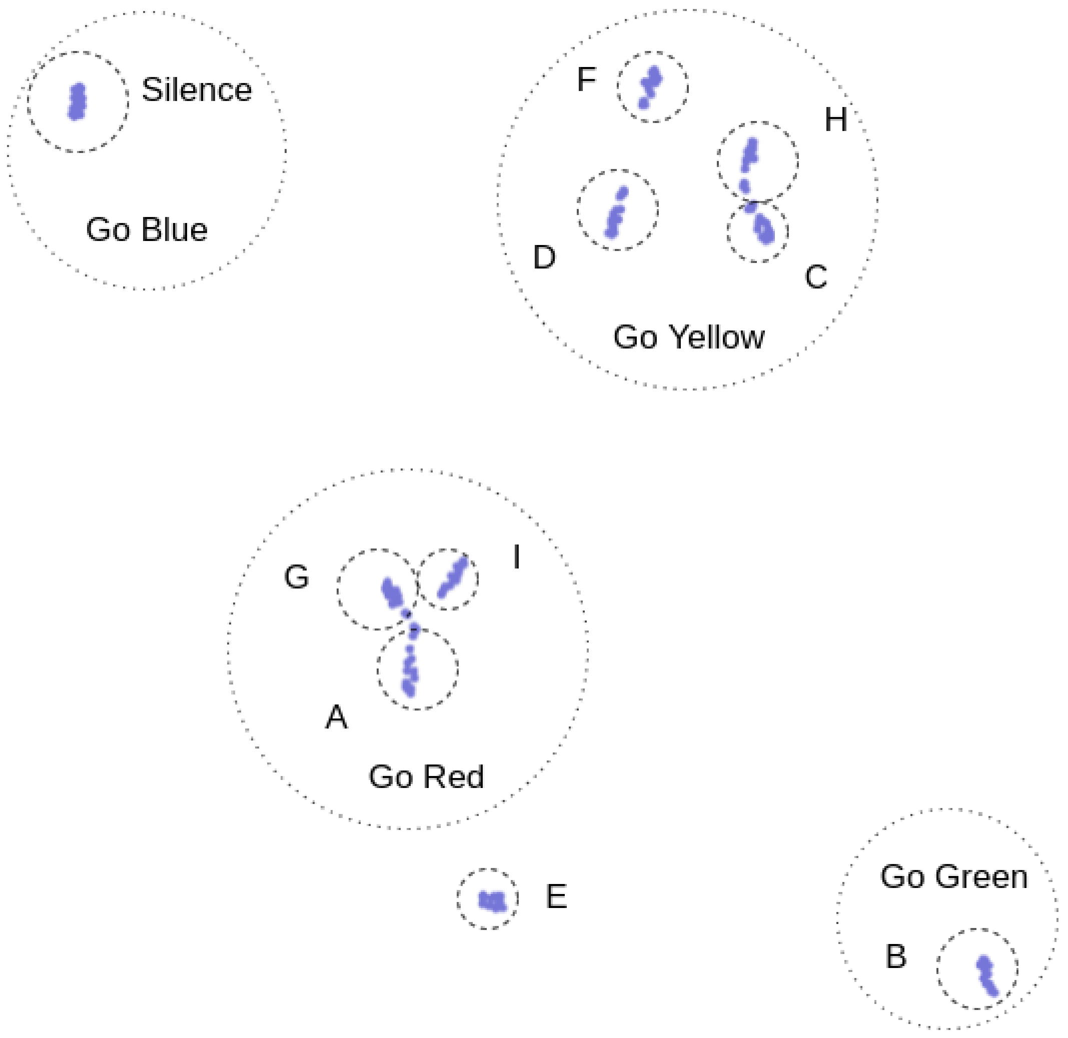

Figure 3.6: Word vector

To further investigate, we zoom in on the ”Go Yellow” cluster and visualize the effect of the agent’s goal color. We display this in figure 3.7. Here, we see that when the personal goal is red, the vectors are away from the center. Moreover, as we go from red -¿ blue -¿ yellow -¿ to green, the clusters get closer to centers. I.e. the symbol D means something much different when the agent’s personal goal is red vs green.

It might be possible that the other agent infers the meaning based on this, and if so, has encoded a lot more meaning than the minimum 4 needed.

3.2.3 Restraining Vocab Size

Thus, the language we have observed has been holistic, or even worse with multiple symbols representing singular meaning. Next, we try to create an artificial bottleneck in form of the maximum types of symbols the agents can emit. We hypothesize that if $vocab_{size} \leq concepts_{size}$, then the agents will be forced to combine the symbols to form higher representations.

I.e., we restrict the vocab size to 2. Therefore, the agents must now come up with a

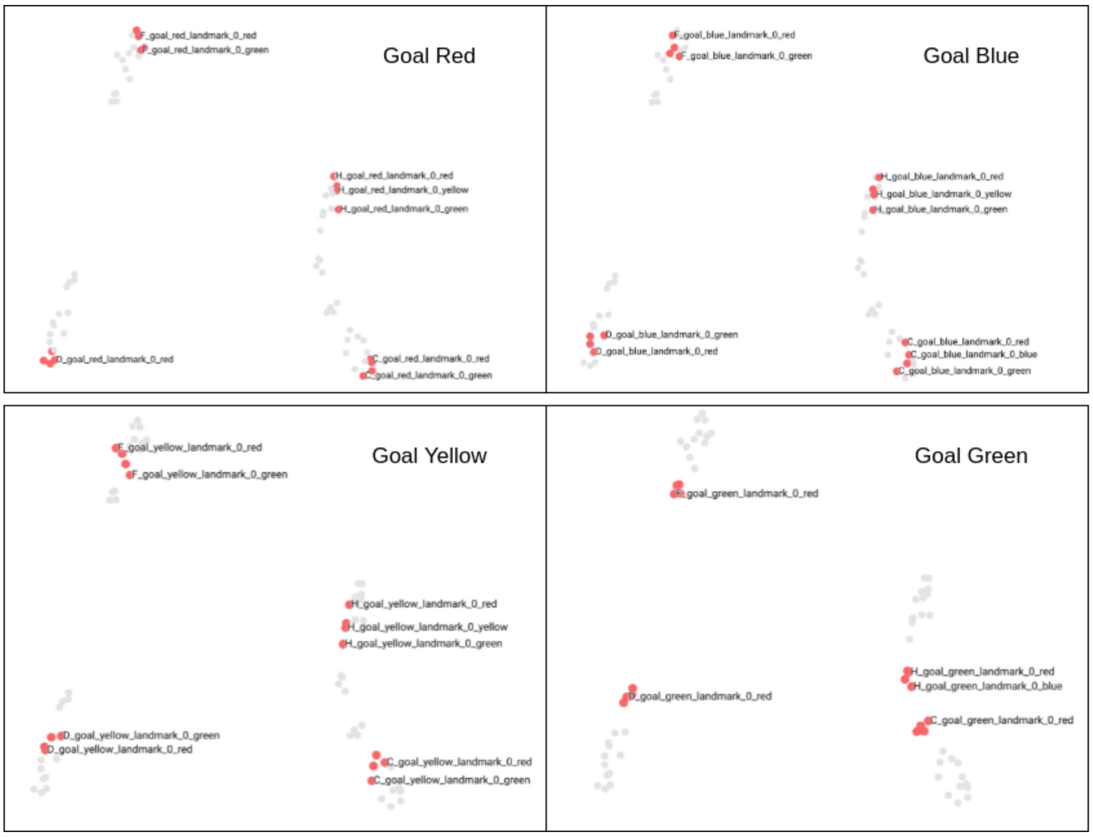

Figure 3.7: context Word vector

language that combines these two symbols. We observe the resulting model for a 3 agent scenario in 3.8. Based on the videos, the first symbol is to communicate which agent is being referred. Since each agent talks to 2 other agents, the 2 symbols at one time step are sufficient. Next, for 4 symbols, agents indicate the desired landmarks. The remaining symbols are emitted to reassure the direction based on the agent observing the agent's position.

Here, we see that the languages formed can be compositional, however require an artificial penalty.

3.2.3.1 Natural language structure and its absence

Previously, we have developed a way for agents to interact with each other in our environemnt 1.1 to make an artificial language. However, we find that the agents always create holisitc languages, a language where every meaning is assigned a unique unit.

Moreover, previous works utilized artificial means to push these languages towards mimicing natural language structure [15], [14], [23]. Most notably, this includes penalizing the vocab size. We demonstrated a similar phenomenon above, when restricting the vocab size to 2. This forced the agent to develop a Morse code like language.

However, as humans we do not receive any direct penalty for a larger vocab size from

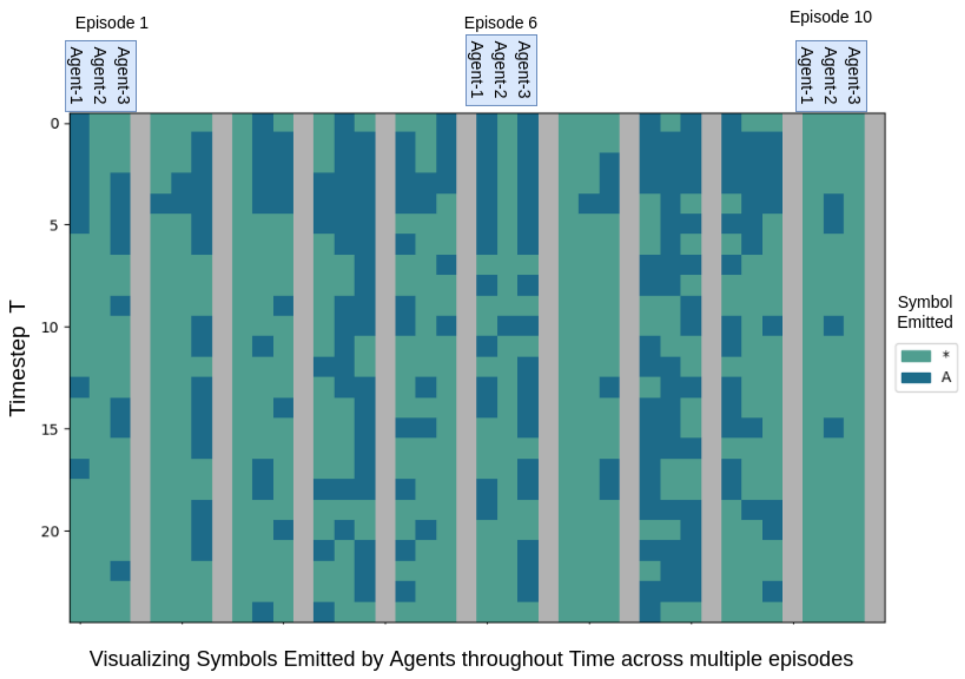

Figure 3.8: Restricting Symbol size to 2. Morse Code like Language

the environment. However, since we do not have an infinite vocab size, we may receive an implicit penalty in the form of being unable to solve a task requiring high vocab size. However, the current models are unable to assign the credit of failure to a large vocab size, as this is a much harder problem.

Next, we explore more natural ways to observe structure in language. It is possible that maybe this emergent structure is completely unintentional.

3.3 Iterated learning paradigm

3.3.1 Introduction

Natural Language exhibits striking structural design features, making it extremely fascinating. Particularly, utterances are constructed out of subparts — phonemes, morphemes, words, phrases — that are reused and recombined in systematic ways.

In addition to exhibiting structure, language is one of a rare set of behaviours that persists through a particular kind of cultural transmission: iterated learning [17].

> The process by which a behaviour arises in one individual through induction on the basis of observations of behaviour in another individual who acquired that behaviour in the same way.

Particularly it argues [17], we induce the particular properties of our language by being exposed to the linguistic behaviour of other individuals in our speech community. Our resulting language in turn leads to linguistic behaviour that shapes the language of further individuals, leading to the possibility of cultural evolution by a process of repeated induction and production of behaviour.

In [19], they survey simulations, mathematical models and experiments all pointing towards the same underlying hypothesis: that the key structural design features of language have their explanation in the fact that language is culturally transmitted in this way [4].

In their previous work [18], perform a series of lab experiments with actual human participants to simulate and demonstrate iterated learning.

3.3.1.1 Experiment1

First, the participants are asked to learn an 'alien' language made up of written labels for visual stimuli. The stimuli are pictures of colored objects in motion, and the labels are sequences of lowercase letters. These sequences are generated and assigned randomly. For training purposes, the language to be learned (a set of string–picture pairs) is divided randomly into 2 sets of approximately equal size: the SEEN set and an UNSEEN set. A participant is trained on the SEEN set.

During subsequent testing, participants are presented with a picture and asked to produce the correct string. Participants are tested on both the SEEN and UNSEEN sets. The first participant is trained on the random language. Subsequent participants are trained on the output of the final testing of the previous participant, which is redivided into new SEEN and UNSEEN sets. The participants' goal is to reproduce the language, not improve on it in it.

Performing this experiment, resulted in languages which were underspecified in nature. I.e., a single word was assigned to multiple meanings of shape and color, rendering it functionally useless.

3.3.1.2 Experiment2

To avoid evolving underspecific languages, they make a small experimental modification.

They 'filter' the SEEN set before each participant's training. If any strings were assigned to more than 1 meaning, all but 1 of those meanings (chosen at random) was removed from the training data. This filtering effectively removes the possibility of the language adapting to be learnable by introducing underspecification: filtering ensures that underspecification is an evolutionary dead-end.

3.3.2 Iterated Learning Simualtion design

They argue, although artificial, is an analogue of a pressure, to be expressive that would come from communicative needs in the case of real language transmission.

Previously, we saw that our agents generated holisitic languages, devoid of structure as observed in natural languages. Believing in the iterated learning hypothesis, and using our environment structure, we can simulate the same experiment described above without the artificial filtering process. Specifically, the agent's goal is not to learn the language, but solve the task of the environment. This is the missing evolutionary filter that no other simulation or lab experiments have previously demonstrated, and will allow us to truly test the iterated learning paradigm.

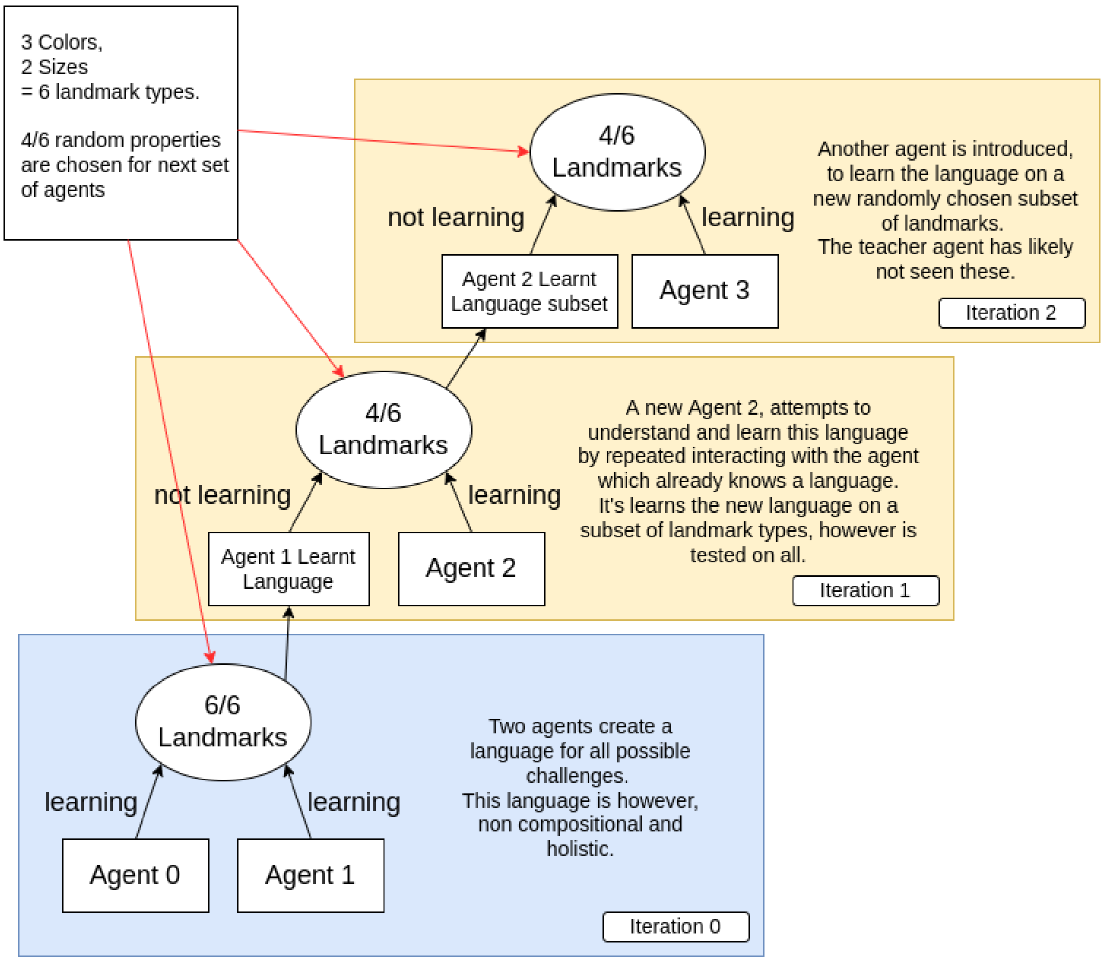
3 Colors,
2 Sizes
= 6 landmark types.
4/6 random properties are chosen for next set of agents
4/6 Landmarks
not learning
learning
Agent 2 Learnt Language subset
Agent 3
Another agent is introduced, to learn the language on a new randomly chosen subset of landmarks. The teacher agent has likely not seen these.
Iteration 2
4/6 Landmarks
not learning
learning
Agent 1 Learnt Language
Agent 2
A new Agent 2, attempts to understand and learn this language by repeated interacting with the agent which already knows a language. It's learns the new language on a subset of landmark types, however is tested on all.
Iteration 1
6/6 Landmarks
learning
learning
Agent 0
Agent 1
Two agents create a language for all possible challenges. This language is however, non compositional and holistic.
Iteration 0

Figure 3.9: Iterated learning Paradigm

In figure 3.9, we layout the general iterated learning paradigm for the environment and agents. Note, this set-up is analogous to the set up in [18], except that the goal of the agents here is to solve the task of reaching the correct landmark, and the goal of language learning is a by-product. This then simulates the pressure to be expressive.

3.3.3 Results

We observed that overtime the overall performance of the agents on the whole test set continues to decrease slowly. Similarly, there were no particular improvements in the type of language produced, vocab size, or number of symbols emitted. Moreover, the agent was unable to perform the task correctly.

Interestingly, we observed that the common cause of failure was that over time the agent would make increasingly more mistakes. Eventually leading to a collapsed model. This is because the previous "teacher" agent, unlike the paired set of symbols from the

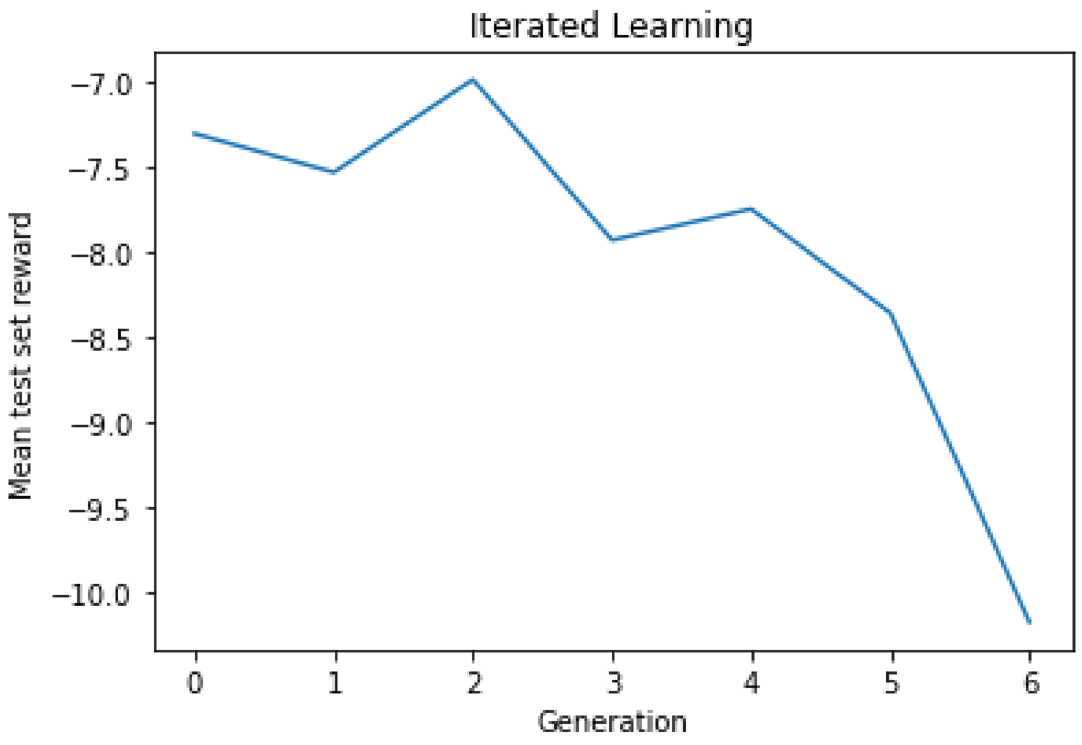

Figure 3.10: Evolution of returns on the Test set over Generations

lab experiments, is crucial to solving the task of the environment. Therefore, at each generation, the teacher agent is worse than the teacher agent at the previous generation. Eventually, leading to a collapsed set of models.

3.3.4 The learning speed dilemma!

However, since we realise that the teacher agent is not adapting at all to the new environment of the next generation, causing it emit incorrect symbols. Moreover, since the teacher agent is not training, it does not optimize for the new environment at all.

This therefore, is not analogous to the need to be expressive that would come from real language transmission.

However, if we were to enable both models to learn at the same rate, it would be no different than training two agents without iterated learning. However, in real life it is said that infants learn at a much faster pace than adults. Therefore, instead of not allowing the teacher agent to learn at all, we set the learning rate of the teacher agent to be 50 times smaller than the learner agent.

3.3.5 Observing the emergence of some compositonality

In 3.11 we see that over generations the return values goes up. More interestingly, these values are for agents which have not see all the properties of the landmarks, are yet able to capture and distinguish well between them.

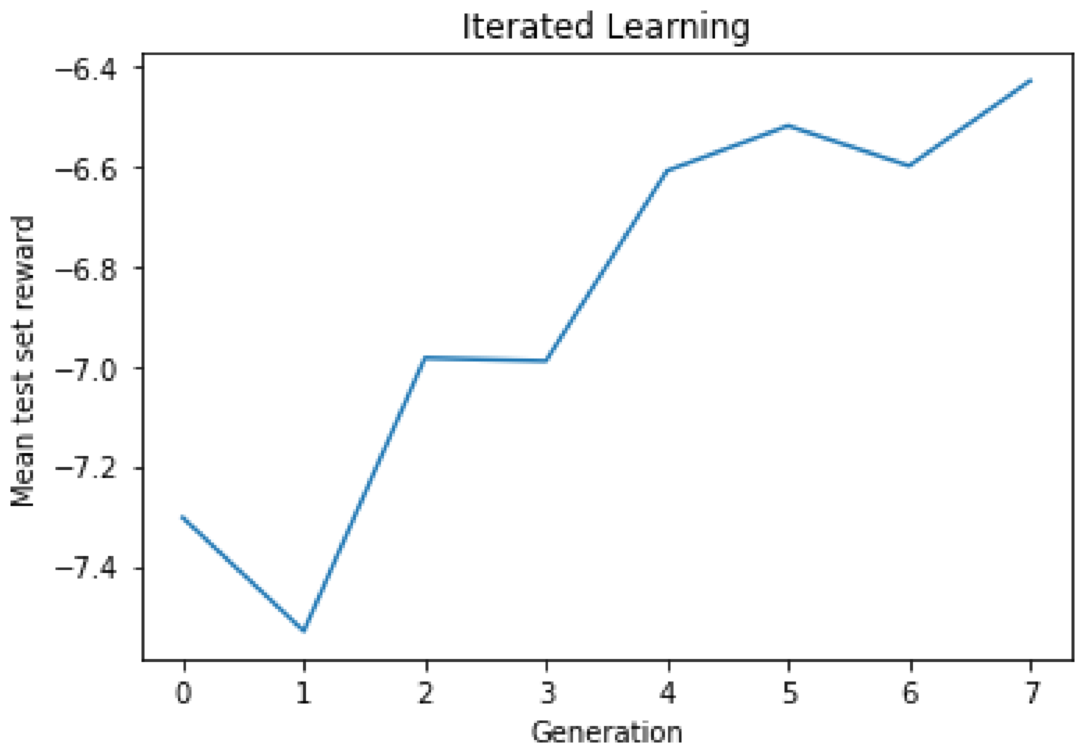

Figure 3.11: Evolution of returns on the **TEST** set over Generations, for different learning rate teacher student models.

I believe that the major difference, between the iterated learning paradigm with no learning, vs slow learning for teacher is that for the first scenario the student has to find patterns. However, if a certain generation has a collapsed model, then there remain no patterns to find. However, for the slow learner case, it helps avoid the bad case of a collapsed model, by allowing the model to always adapt at a slower pace. Moreover, by learning slowly with a new agent who doesn't follow the same language, the teacher is also forced to optimize in a direction to help the learner adapt faster.

Chapter 4

Conclusions

We created a new set of environments based on Open-AI's MPE engine for language evolution. Following the utilitarian paradigm, the agents are given tasks to solve in the environment. These tasks, however, required communication. We reviewed various existing works on MARL and designed a language learner agent capable of solving tasks.

We showed that in agreement to [21] structure in artifical language generation through RL can only happen through a forced set of rules, like vocab size restrictions. Otherwise, the languages generated are hollisitc. However, inspired by iterated learning, we showed that maybe the structure of language is the result of cultural transmission. Moreover, we are the first to show that this paradigm of iterated learning holds without artificial filtering rules. We are also the first to simulate iterated learning in a continuous time environment, where the objective is not language. Rather, precisely due to these properties of our environment, we are able to enforce the pressure of expressivity on agents without artificial means. As a result, the only evolutionary pathway for the model is to develop a structure as seen in natural language.

4.0.1 Future Work

We have shown promise through simulation of language evolution through the utilitarian paradigm. However, developing automated tools to analyze the strucutre of large vocab foriegn languages, like those of the agents, will greatly help us understand the merits.

We also believe, that the current MARL algorithms are quite sample inefficient. If we wish to interact and collaborate with these agents, we must ensure that they can learn faster. We believe, it's promising to build algorithms which are sample efficient at first to build a better learning bias. These can then be finetuned to the same environment in a much more sample efficient way.

Lastly, all of our experiments have been performed on an extremely toy example compared to natural language. Simulating more concepts, potentially through architectures utilizing vision can help strength our findings.

Bibliography

[1] Adrian K. Agogino and Kagan Tumer. Unifying temporal and structural credit assignment problems. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '04, page 980–987, USA, 2004. IEEE Computer Society.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[3] Justin Boyan and Andrew Moore. Generalization in reinforcement learning: Safely approximating the value function. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994.

[4] Henry Brighton, Kenny Smith, and Simon Kirby. Language as an evolutionary system. *Physics of Life Reviews*, 2(3):177–226, 2005.

[5] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge?, 2020.

[6] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020.

[7] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[8] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients, 2017.

[9] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate to solve riddles with deep distributed recurrent q-networks, 2016.

[10] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause,

editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 10–15 Jul 2018.

[11] Jon Gauthier and Igor Mordatch. A paradigm for situated and goal-driven language learning, 2016.

[12] Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored mdps. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

[13] Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols, 2017.

[14] Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols, 2017.

[15] Pieter Abbeel Igor Mordatch. Emergence of grounded compositional language in multi-agent populations, 2018.

[16] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation, 2018.

[17] S. Kirby. Spontaneous evolution of linguistic structure - an iterated learning model of the emergence of regularity and irregularity. *IEEE Transactions on Evolutionary Computation*, 5(2):102–110, 2001. Cited By :237.

[18] Simon Kirby, Hannah Cornish, and Kenny Smith. Cumulative cultural evolution in the laboratory: An experimental approach to the origins of structure in human language. *Proceedings of the National Academy of Sciences*, 105(31):10681–10686, 2008.

[19] Simon Kirby, Tom Griffiths, and Kenny Smith. Iterated learning and the evolution of language. *Current Opinion in Neurobiology*, 28:108–114, 10 2014.

[20] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[21] Satwik Kottur, José Moura, Stefan Lee, and Dhruv Batra. Natural language does not emerge 'naturally' in multi-agent dialog. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2962–2967, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[22] Guillaume J. Laurent, Laëtitia Matignon, and N. Le Fort-Piat. The world of independent learners is not markovian. *Int. J. Know.-Based Intell. Eng. Syst.*, 15(1):55–64, jan 2011.

[23] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language, 2017.

[24] David Lewis. *Convention: a philosophical study*. Cambridge, Harvard university press, 1969.

[25] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In William W. Cohen and Haym Hirsh, editors, *Machine Learning Proceedings 1994*, pages 157–163. Morgan Kaufmann, San Francisco (CA), 1994.

[26] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments, 2020.

[27] Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. Contrasting centralized and decentralized critics in multi-agent reinforcement learning. 2021.

[28] Laetitia Matignon, Guillaume J. Laurent, and Nadine Le Fort-Piat. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1):1–31, 2012.

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[30] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games, 2015.

[31] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games, 2017.

[32] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning, 2018.

[33] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2015.

[34] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2015.

[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[36] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks, 2018.

[37] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation, 2016.

[38] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning, 2017.

[39] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[40] J. K Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sulivan, Luis Santos, Rodrigo Perez, Caroline Horsch, Clemens Dieffendahl, Niall L Williams, Yashas Lokesh, Ryan Sullivan, and Praveen Ravi. Pettingzoo: Gym for multi-agent reinforcement learning. *arXiv preprint arXiv:2009.14471*, 2020.

[41] Kagan Tumer and Adrian Agogino. Distributed agent-based air traffic flow management. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '07, New York, NY, USA, 2007. Association for Computing Machinery.

[42] Ermo Wei and Sean Luke. Lenient learning in independent-learner stochastic cooperative games. *Journal of Machine Learning Research*, 17(84):1–42, 2016.

[43] DAVID H. WOLPERT and KAGAN TUMER. *Optimal Payoff Functions for Members of Collectives*, pages 355–369.

[44] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games, 2021.