Natural Language Does Not Emerge 'Naturally' in Multi-Agent Dialog

Satwik Kottur[1] and **José M.F. Moura**[1] and **Stefan Lee**[2,3] and **Dhruv Batra**[3,4]
[1]Carnegie Mellon University, [2]Virginia Tech, [3]Georgia Tech, [4]Facebook AI Research

The supplement is organized as follows:

- Sec. 1 overviews all the settings characterizing communication language between the agents, along with detailed learning equations and implementation details,
- Sec. 2 gives details of learnt grounding in two important settings,
- Sec. 3 describes the evolution of language in detail, beginning with the structure and construction of dialog trees, and continues to build on top to explain the procedure to obtain a timeline of how the agents learn grounding for symbols at intermediate stages of training.

1 Overview

All our findings are summarized in Tab. 1. We show some example conversations for: (a) Overcomplete vocabularies setting (Sec.4.1 in main) in Fig. 1, (b) Attribute and Value Vocabulary setting (Sec.4.2 in main) in Fig. 2, (c) Memoryless A-BOT, Minimal Vocabulary setting (Sec.4.3 in main) in Fig. 3. We re-emphasize that compositionality in language does not emerge without the need for it. If multiple optimal policies are attainable, the model picks policies that are easier to learn, *e.g.*, given a large vocabulary, the model enumerates all possible instances, than learn to compose instances as combinations of individual attributes (Das et al., 2017).

Learning Policies with REINFORCE. To train these agents, we update policy parameters θ_Q and θ_A using the popular REINFORCE (Williams, 1992) policy gradient algorithm. Note that while the game is fully-cooperative, we do not assume full observability of one agent by another, opting instead to treat each agent as part of the stochastic environment when updating the other. We will now derive the parameter gradients for our setup.

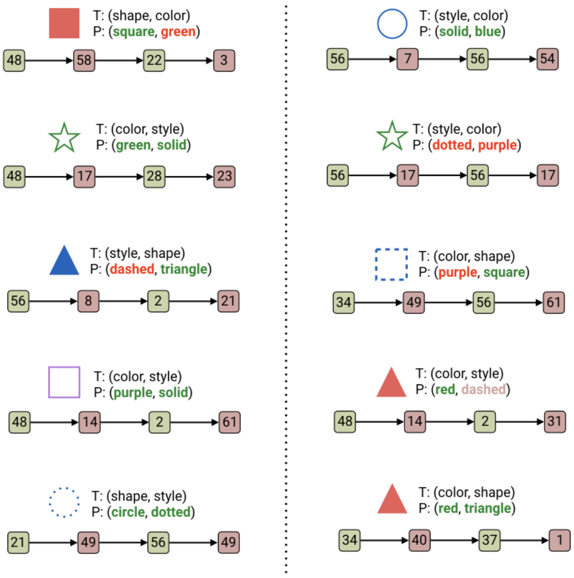

Figure 1: Example dialogs for Overcomplete vocabularies setting (Sec.4.1 in main). As $|V_Q| = |V_A| = 64$, we denote the tokens with just numbers where color of boxes indicates whether they were uttered by A-BOT or Q-BOT. The learnt mapping is highly non-human intuitive and non-compositional.

Recall that our agents take actions – utterances (q_t and a_t) and attribute prediction ($\hat{w}_G$) – and our objective is to maximize the expected reward under the agents' policies:

$$\max_{\theta_A, \theta_Q} J(\theta_A, \theta_Q) \quad \text{where,} \tag{1a}$$

$$J(\theta_A, \theta_Q) = \mathop{\mathbb{E}}_{\pi_Q, \pi_A} \left[R\big(\hat{w}^G, w^G\big) \right] \tag{1b}$$

Though the agents receive the reward at the end of gameplay, all intermediate actions are assigned the same reward R. Following the REINFORCE algorithm, we write the gradient of this expectation as an expectation of policy gradients. For θ_Q, we derive this explicitly at a time step t:

Setting	Vocab.		Memory		Seen (%)		Unseen (%)		Characteristics
	V_Q	V_A	A	Q	Both	One	Both	One	
Overcomplete (Sec.4.1 main)	64	64	✓	✓	100	100	25.6	79.5	• Non-compositional language • Q-BOT insignificant • Inconsistent A-BOT grounding across rounds • Poor generalization to unseen instances
Attr-Value (Sec.4.2 main)	3	12	✓	✓	100	100	38.5	88.4	• Non-compositional language • Q-BOT uses one round to convey task • Inconsistent A-BOT grounding across rounds • Poor generalization to unseen instances
NoMem-Min (Sec.4.3 main)	3	4	✗	✓	100	100	**74.4**	**94.9**	• Compositional language • Q-BOT uses both rounds to convey task • Consistent A-BOT grounding across rounds • Good generalization to unseen instances

Table 1: Overview of settings we explore to analyze the language learnt by two agents in a cooperative game, Task & Talk. Last two columns measure generalization in terms of prediction accuracy of **both** or at least **one** of the attribute pair, on a held-out test set containing unseen instances.

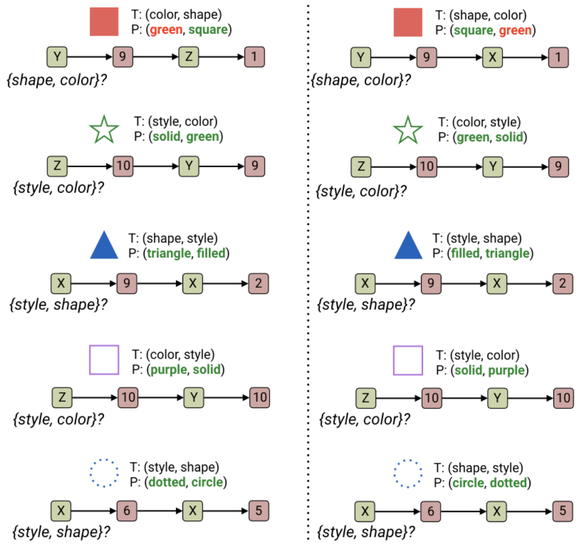

Figure 2: Example dialogs for Attribute and Value vocabulary setting (Sec.4.2 in main). We show both the symmetric tasks for each image to note the difference in the language between the agents. As seen here, Q-BOT learns to map symmetric tasks in an order-agnostic fashion, and uses only the first token to convey task information to A-BOT.

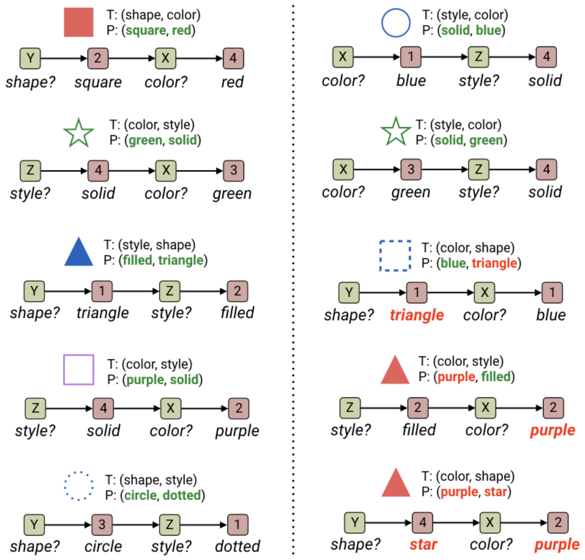

Figure 3: Example dialogs for Memoryless A-BOT and Minimal vocabulary setting (Sec.4.3 in main). Both Q-BOT and A-BOT learn consistent and grounded language, which is shown below each utterance. We also show negative examples on unseen instance, where the model gets either one or both attributes incorrect in the pair. Although the predicted attribute value is incorrect, note that it is still of the right kind (a color attribute for *color*, but not a style/shape attribute). Exact details of the mapping are present in Sec. 2

$$
\begin{aligned}
\nabla_{\theta_Q} J &= \nabla_{\theta_Q} \left[\underset{\pi_Q, \pi_A}{\mathbb{E}} \left[R(\hat{w}^G, w^G) \right] \right] \\
&= \nabla_{\theta_Q} \left[\sum_{q_t, a_t} \pi_Q \left(q_t | s_{t-1}^Q \right) \pi_A \left(a_t | s_t^A \right) R(.) \right] \\
&= \sum_{q_t, a_t} \pi_Q \left(q_t | s_{t-1}^Q \right) \nabla_{\theta_Q} \log \pi_Q \left(q_t | s_{t-1}^Q \right) \pi_A \left(a_t | s_t^A \right) R(.) \\
&= \underset{\pi_Q, \pi_A}{\mathbb{E}} \left[R(.) \, \nabla_{\theta_Q} \log \pi_Q \left(q_t | s_{t-1}^Q \right) \right] \qquad (2)
\end{aligned}
$$

Similarly, gradient w.r.t. θ_A, *i.e.*, $\nabla_{\theta_A} J$ will be:

$$
\nabla_{\theta_A} J = \underset{\pi_Q, \pi_A}{\mathbb{E}} \left[R(.) \, \nabla_{\theta_A} \log \pi_A \left(a_t | s_t^A \right) \right] \qquad (3)
$$

As is standard practice, we estimate these expectations with sample averages – sampling an environment (object instance and task), sampling a dialog between Q-BOT and A-BOT, culminating in a prediction from Q-BOT and the received reward. The REINFORCE update rule above has an intuitive interpretation – an *informative* dialog (q_t, a_t) that leads to positive reward will be made more probable (positive gradient), while a poor exchange leading to negative reward will be pushed down (negative gradient).

Implementation Details. All our models are implemented using the Pytorch[1] deep learning framework. To represent instances, we learn a 20 dimensional embedding for every possible attribute values and concatenate the three instance attributes to obtain a final instance representation of size 60. Tokens from V_Q and V_A are encoded as one-hot vectors and then embedded into 20 dimension vectors. Both A-BOT and Q-BOT learn their own token embeddings without sharing. The listener networks in both agents are implemented as LSTMs with a hidden layer size of 50 dimensions. All modules within an agent are initialized using the Xavier method (Glorot and Bengio, 2010).

We use 1000 episodes of two-round dialogs to compute policy gradients, and perform updates according to Adam optimizer (Kingma and Ba, 2015), with a learning rate of 0.01. Furthermore, gradients are clipped at $[-5.0, 5.0]$. For faster convergence, 80% of train episodes for the next iteration are from instances misclassified by the current network, while randomly sampling the remaining from all instances. Our code is publicly available[2].

2 Emergence of Grounded Language

We list out the language grounding learnt by both Q-BOT and A-BOT for two settings:

Attribute and Value Vocabulary. We observe that Q-BOT encodes tasks in an order-agnostic fashion using first round token only, as: *(color, shape),(shape, color)* $\rightarrow$ Y, *(color, style),(style, color)* $\rightarrow$ Z, and *(style, shape),(shape, style)* $\rightarrow$ X.

Memoryless A-BOT, Minimal Vocabulary. In case of Memoryless A-BOT and limited vocabulary for the agents, $|V_Q|$=$|\{X, Y, Z\}|$=3, $|V_A|$=$|\{1, 2, 3, 4\}|$=4, compositional groundings emerge that are consistent across rounds, as shown in Tab. 2.

↓ V_A	**Attributes**		
V_Q →	*color* *X*	*shape* *Y*	*style* *Z*
1	*blue*	*triangle*	*dotted*
2	*purple*	*square*	*filled*
3	*green*	*circle*	*dashed*
4	*red*	*start*	*solid*

(a) Grounding for A-BOT ‘s responses given Q-BOT ‘s query

Task	q_1	q_2
(color, shape)	*Y*	*X*
(shape, color)	*Y*	*X*
(shape, style)	*Y*	*Z*
(style, shape)	*Y*	*Z*
(color, style)	*Z*	*X*
(style, color)	*X*	*Z*

(b) Grounding for Q-BOT ‘s responses given a task

Table 2: Emergence of compositional grounding for language learnt by the agents. **A-BOT** (Tab. 2a) learns meanings that are consistent across rounds, depending on the query attribute. Token grounding for **Q-BOT** (Tab. 2b) depends on the task at hand. Notice that although compositional, Q-BOT does not necessarily query attribute in the order of task, but rather re-arranged accordingly at prediction time as it contains memory.

From Tab. 2, we can predict the plausible dialog between the agents for any unseen instance + task combination. Notice that this is possible only due to the compositionality in the emergent language between the two agents. As as example, consider solving the task *(color, shape)* for an unseen instance *(purple, circle, dotted)*. We can read off the likely dialog path as: $(q_1{=}Y)\rightarrow(a_1{=}3)\rightarrow(q_2{=}X)\rightarrow(a_2{=}2)$.

---

[1] `github.com/pytorch/pytorch`

[2] `github.com/batra-mlp-lab/lang-emerge`

3 Evolution of Language

As demonstrated in the main paper, even though compositional language is one of the optimal policies, the agents tend to learn other equally useful forms of communication. Thus, compositional language does not naturally emerge without an explicit need for it. Even in situations where compositionality does emerge, perhaps it is more interesting to analyze the process of emergence than the learnt language itself. Therefore, we present such a study that explicitly identifies *when* each symbol has been grounded by the agents in the training timeline, along with implications thereof on the performance on Task & Talk game.

3.1 Dialog Trees

When two agents–Q-BOT and A-BOT–converse with each other, they can be seen as traversing through a dialog tree, a subtree of which is depicted in Fig. 4. Simply put, a dialog tree is an enumeration of all possible dialogs represented in the form of tree, with levels of the tree corresponding to the round of interaction. To elaborate, consider a partial dialog tree for *(shape, color)* task shown in Fig. 4 for the setting in Sec.4.3 of main. For Q-BOT's first token $q_1 = Y$, A-BOT has $|V_A| = 4$ plausible replies shown as a 4-way branch off. In general, the dialog tree for Task & Talk contains a total of $|V_Q|^2|V_A|^2$ leaves and is 4 levels deep. We use the dialog between the agents to descend and land in one of these leaves.

Dialog trees offer an interesting alternate view of our learning problem. The goal of learning communication between the two agents can be equivalently seen as mapping *(instance, task)* pairs to one of the dialog tree leaves. Each leaf is labeled with an attribute pair used to accomplish the prediction task. For example, if solving *(shape, color)* for *(blue, triangle, solid)* results in the dialog $Y \rightarrow 1 \rightarrow X \rightarrow 1$, we descend the dialog tree along the corresponding path and assign the tuple *(blue, triangle, solid, shape, color)* to the resulting leaf. In case of a compositional, grounded dialog, all tuples of the form *(blue, triangle, $*$, shape, color)* would get mapped to the same leaf, which can then be labeled as *(triangle, blue)* to successfully solve the task. Note the wildcard *style* attribute in the tuple above, as it is irrelevant for this particular task.

In the following section, we use dialog trees to explore the evolution of language as learnt by the two agents in the memoryless A-bot, minimal vocabulary setting in Sec.4.3 of main.

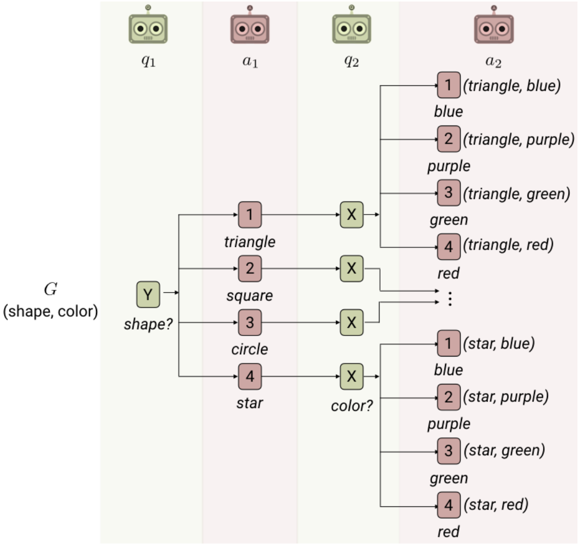

Figure 4: Dialog tree for memoryless A-BOT and minimal vocabulary setting (Sec.4.3), shown only for one task *(shape, color)*. Every dialog between the agents results in a tree traversal beginning from the root, *e.g.*, $Y \rightarrow 1 \rightarrow X \rightarrow 1$ lands us in the top-right leaf. See text for more details.

3.2 Evolution Timeline

To gain further insight into the languages learned, we create a *language evolution* plot shown in Fig. 5. Specifically, at regular intervals during policy learning, we construct dialog trees. At some point in the learning, the nodes in the tree become and stay 'pure' (all *(instance, task)* at the node are identical), at which point we can say that the agents have learned this dialog subsequence. Fig. 5 depicts a timeline of concepts learned at various nodes of the trees during training. We next describe the procedure to identify when a particular 'concept' has been grounded by the agents in their language.

Construction. After constructing dialog trees at regular intervals, we identify 'concepts' at each node/leaf using the dialog tree of the completely trained model, which achieves a perfect accuracy on train set. A concept is simply the common trend among all the *(instance, task)* tuples either assigned to a leaf or contained within the subtree with a node as root. To illustrate, the concept of the top right leaf in Fig. 4 is *(blue, triangle, $*$, shape, color)*, *i.e.*, all instances assigned to that leaf for *(shape, color)* task are blue triangles. Next, given a resultant concept for each of the node/leaf, we backtrack in time and check for the first occurrence when only tuples which satisfy

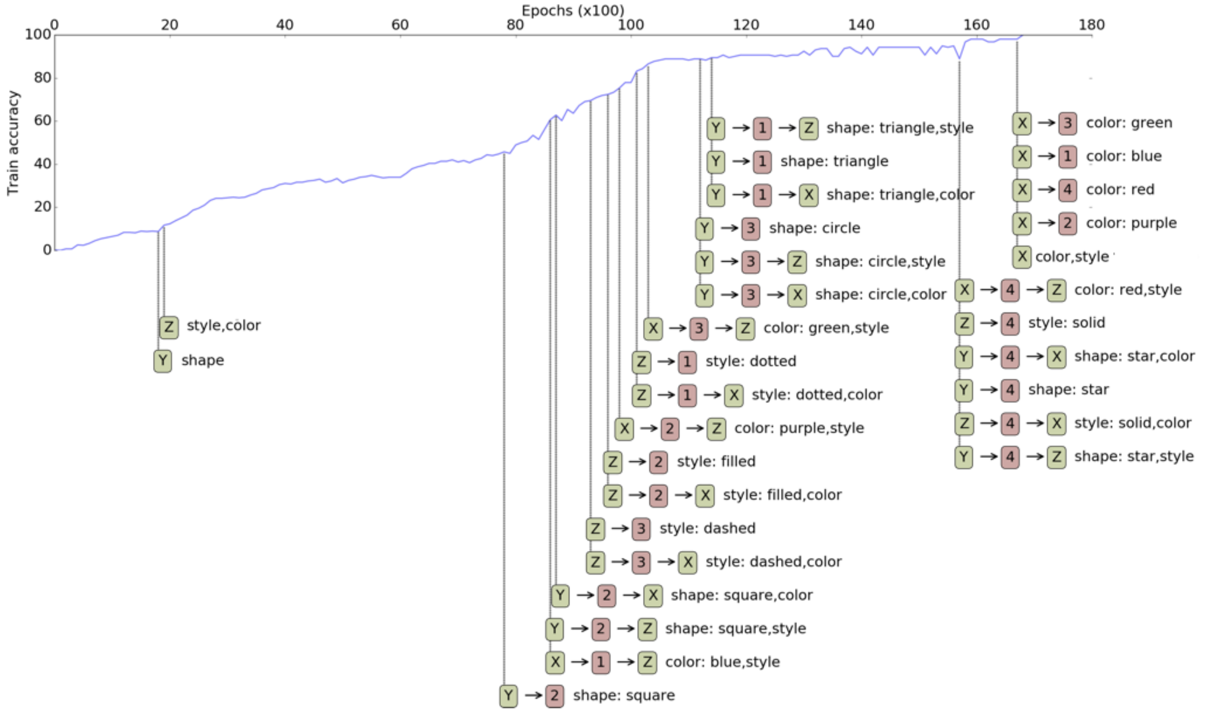

Figure 5: Evolution of Language: timeline shows groundings learned by the agents during training, overlaid on the accuracy. Note that Q-BOT learns encodings for all tasks early (around epoch 20) except *(style, color)*. Improvement in accuracy is strongly correlated with groundings learnt.

the corresponding concept are assigned to that particular node/leaf. In other words, we compute the earliest time when a node/leaf is ‘pure’ with respect to its final learned concept. Finally, we plot these leaves/nodes and the associated concept with their backtracked time to get Fig. 5.

Observations. We highlight the key observations from Fig. 5 below:

(a) The agents ground most of the tasks initially at around epoch 20. Specifically, Q-BOT assigns Y to both *(shape, style), (style, shape), (shape,color)* and *(color, shape)*, while *(color, style)* is mapped to Z. Hence, Q-BOT learns its first token very early into the training procedure at around 20 epochs.

(b) The only other task *(style, color)* is grounded towards the end (around epoch 170) using X, leading to an immediate convergence.

(c) We see a strong correlation between improvement in performance and when agents learn a language grounding. In particular, there is an improvement from 40% to 80% within a span of 25 epochs where most of the grounding is achieved, as seen from Fig. 5.

References

Abhishek Das, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra. 2017. Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning. *arXiv preprint arXiv:1703.06585* .

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

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