Enhance the Compositionality of Emergent Language by Iterated Learning

Yi Ren, Shangmin Guo, Serhii Havrylov, Shay B. Cohen, Simon Kirby
University of Edinburgh
Y.Ren-18@sms.ed.ac.uk

Abstract

Compositionality, which enables the natural language to represent complex concepts using the combination of simple parts, allows us to convey an open-ended set of messages using limited vocabulary and grammar rules. Hence researchers expect the emergent communication protocol invented by the neural agents also have similar properties. Inspired by the language evolutionary procedure and the account proposed by the linguists, we propose an effective neural iterated learning algorithm to enhance the compositionality of the emergent language. We also propose a probabilistic explanation to articulate the mechanisms of the algorithm, which can also interpret many findings in related works. Using experimental results and ablation studies, our proposed algorithm are proved to be effective in facilitating the emergence of high-compositional languages, which can performs much better than low-compositional languages in zero-shot experiments.

1 Introduction

Natural language understanding (NLU), which aims at teaching computers to understand human language, plays a crucial role in artificial intelligence systems. Inspired by the origin of human language, grounded language learning gradually attracts more attention in recent years. Different from traditional NLU systems, the works in this direction focus on the pragmatics (Clark, 1996) of natural language: they put some neural agents into a game and encourage these agents to communicate with each other to accomplish specific tasks. During this process, the agents may build mappings between concepts and message symbols. Such mappings are usually called emergent language.

So far, lots of notable works (e.g., (Havrylov and Titov, 2017; Mordatch and Abbeel, 2018; Kottur et al., 2017; Foerster et al., 2016)) demonstrate that in many game settings, the agents can use their invented emergent communication protocol to exchange information. Although how to design games to stimuli the language emergence is still in debate, most of the researchers reach a consensus that we should encourage such languages to share similarities with natural language. Among all the properties of human language, compositionality is considered to be a critical one, because it enables us to represent complex concepts by combining several simple concepts. Study in this direction is still in its early stage, even though some works have already demonstrated that by properly choosing the maximum message length and vocabulary size, the agents can invent compositional language similar to natural language, e.g., (Li and Bowling, 2019; Lazaridou et al., 2018; Cogswell et al., 2019).

On the other hand, evolutionary linguists have already studied the origins and the measurements of language compositionality for decades (Kirby and Hurford, 2002; Brighton and Kirby, 2006; Kirby et al., 2014, 2015). They proposed a cultural evolutionary account of the origins of compositionality and designed a framework called iterated learning to simulate the language evolution process. However, the works mentioned above are mainly verified on Bayesian agents and rational volunteers. For the Bayesian agent's case, researchers design a prior probability that favors the high-compositional languages. For the rational volunteer's case, the participants also favor those high-compositional

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

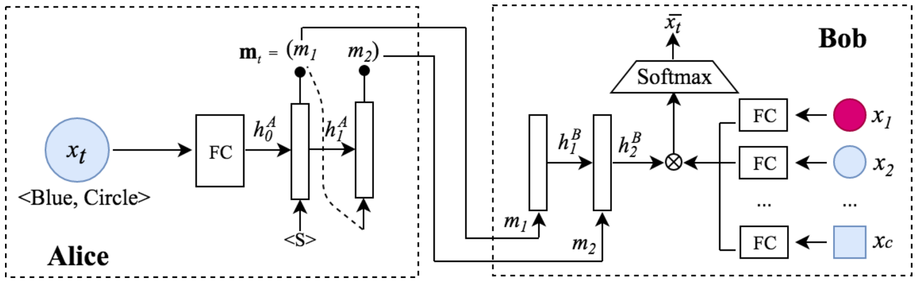


Figure 1: Referential communication game and architectures of the agents.

languages because the human brain prefers the highly-structured language. As such a tendency is not apparent in neural agents, directly applying their proposed framework is not straightforward.

Hence in this project, we propose a three-phase neural iterated learning algorithm to encourage the emergence of high-compositional language. We also propose a probabilistic model to track the distribution of candidate languages in the search space. Such a model can correctly explain why the method applied in the present works can slightly enhance the compositionality of the emergent language. Using a simple referential game (Lewis, 2008), the experimental results prove that our algorithm can significantly encourage the dominance of high-compositional language, which is more similar to natural language and generalize better than low-compositional language.

2 System Model

We analyze a typical object selection game, in which a speaking agent (Alice) and a listening agent (Bob) must cooperate to accomplish a task. In each round of the game, we show Alice a target object x_t selected from an object space $\mathcal{X}$ and let it send a discrete-sequence message $\mathbf{m}_t$ to Bob. We then show Bob c different objects (x_t must be one of them) and use $x_1, ..., x_c \in \mathcal{X}$ to represent these candidates. Based on the message received from Alice, Bob must choose the correct object that Alice refers, from c candidates. If Bob's selection is correct, then both Alice and Bob will get rewards denoted by $r = 1$, otherwise, they will get $r = 0$. We will shuffle the objects and randomly select candidates each round to avoid the agents building mappings between the objects and their order. In our game, each object in $\mathcal{X}$ has two attributes ,i.e., color and shape, and each attribute has eight possible values. We encode each attribute as a one-hot vector and concatenate two one-hot vectors to represent one object. The message delivered by Alice is a fixed-length discrete sequence $\mathbf{m} = (m_1, ..., m_{N_L}) \in \mathcal{M}$, in which each m_i is selected from a fixed size vocabulary V.

The architectures of Alice and Bob are illustrated in Figure 1, which is similar to that studied in (Havrylov and Titov, 2017). Alice first applies a multi-layer perceptron (MLP) to encode x_t to an embedding, then feeds it to an LSTM to generate message $\mathbf{m}_t$. Bob uses an LSTM to read the message and applies the MLPs to encode $x_1, ..., x_c$. Bob then takes the dot product between hidden states of LSTM and MLPs and let the product pass a softmax layer. Using the REINFORCE algorithm (Williams, 1992), the gradients of the objective function $J(\theta_A, \theta_B)$ is:

$$\nabla_{\theta_A} J = \mathbb{E}\left[R(\bar{x_t}, x_t)\nabla \log p_A(\mathbf{m}_t|x_t)\right] + \lambda_A \nabla H[p_A(\mathbf{m}_t|x_t)] \quad (1)$$
$$\nabla_{\theta_B} J = \mathbb{E}\left[R(\bar{x_t}, x_t)\nabla \log p_B(\bar{x_t}|\mathbf{m}_t, x_1, ..., x_c)\right] + \lambda_B \nabla H[p_B(\bar{x_t}|\mathbf{m}_t, x_1, ..., x_c)], \quad (2)$$

where $R(\bar{x_t}, x_t) = \mathbb{1}(\bar{x_t}, x_t)$ is the reward function, $H[\cdot]$ is the standard entropy function, and $\lambda_A, \lambda_B > 0$ controls the extent of regularization.

Throughout this paper, we use topological similarity (proposed in (Brighton and Kirby, 2006)) and the zero-shot performance as the intrinsic and extrinsic measurements of compositionality, respectively. The topological similarity, denoted as ρ, is defined using the correlation coefficient between every element pairs in $\mathcal{X}$ to the corresponding pairs in $\mathcal{M}$. The zero-shot performance is calculated by testing the performance of the trained model on a held-out data set (the size of validation set is roughly 15% of the training set).

Algorithm 1 Neural Iterated Learning Algorithm.

```
1: Randomly initialize $D_0$
2: for $i = 1, 2, ..., I$ do
3:     Re-initialize Alice and Bob, get Alice$_i$ and Bob$_i$
4:     // ======= Learning Phase =======
5:     for $i_a = 1, 2, ..., I_a$ do
6:         Randomly sample pairs from $D_{i-1}$ and train Alice$_i$ using them
7:     end for
8:     for $i_b = 1, 2, ..., I_b$ do
9:         Alice$_i$ generates message based on input objects
10:        Bob$_i$ receives message and select the target
11:        Bob$_i$ update its parameters if reward=1
12:    end for
13:    // ======= Game Playing Phase =======
14:    for $i_g = 1, 2, ..., I_g$ do
15:        Alice$_i$ generates message based on input objects
16:        Bob$_i$ receives message and select the target
17:        BOTH Alice$_i$ and Bob$_i$ update parameters if reward=1
18:    end for
19:    // ======= Sampling Phase =======
20:    for $i_s = 1, 2, ..., I_s$ do
21:        Generate object-message pairs by feeding all objects to Alice$_i$ and save them to data set $D_i$
22:    end for
23: end for
```

3 Neural Iterated Learning

The iterated learning framework is proven to be effective in experiments with both Bayesian agents and human volunteers, but directly applying it to the neural-agent-based games is not that trivial. The first obstacle is that we cannot directly feed the prior probability, which favors the high-ρ language, to the neural network. Furthermore, as Alice and Bob usually have different architectures in most of the neural-agent-based games, the pre-train procedure in the learning phase should be carefully designed. Hence in this section, we propose a neural iterated learning algorithm (so as a probabilistic explanation) which can overcome these obstacles.

The algorithm has three phases: learning phase (or pre-training phase), game playing phase, and sampling phase. At the beginning of each generation, all the agents are randomly re-initialized. Then Alice and Bob will be pre-trained using the data generated by their predecessors, i.e., D_{i-1}. As Alice and Bob have different structures, their pre-training is divided into two parts. After that, Alice and Bob will play the game for I_g rounds, during which they will fine-tune their language to be accurate. Finally, we will feed all objects to Alice and let it generate the corresponding messages. The generated object-message pairs are stored as D_i for the learning phase in the next generation.

As Alice has a neural network which can map any object $x_t \in \mathcal{X}$ to a message $\mathbf{m}_t \in \mathcal{M}$, the output can represent the posterior distribution of the message given the input object, i.e., $P(\mathbf{m}_t|D, x_t)$, where D is the training samples observed by Alice. As a language is defined as a mapping function from all objects in $\mathcal{X}$ to their corresponding messages in $\mathcal{M}$, we can calculate the posterior probability of any language by:

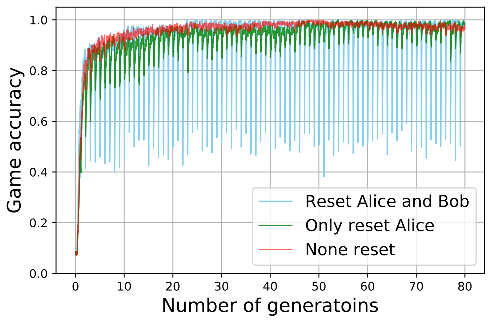
$$P(\mathcal{L}|D) = P(\mathbf{m}_1, ...|D, x_1, ...) = \prod_{\langle x_t, \mathbf{m}_t \rangle \in \mathcal{L}} P(\mathbf{m}_t|D, x_t). \tag{3}$$

Hence we can directly sample languages by feeding all possible objects to Alice and get the corresponding message by sampling the output characters following the posterior distribution (i.e., the output of the softmax layer). We can then calculate the posterior distribution of language samples with different values of ρ, i.e., $P\left(\rho(\mathcal{L})|D_{i-1}\right)$, to better understand how the lanugage evolves during iterated learning. Such a language sampling explanation can correctly interpret why choosing smaller vocabulary size and message length, which are mentioned in many related works, can slightly enhance the compositionality of the emergent language. We will provide the details in the appendix.

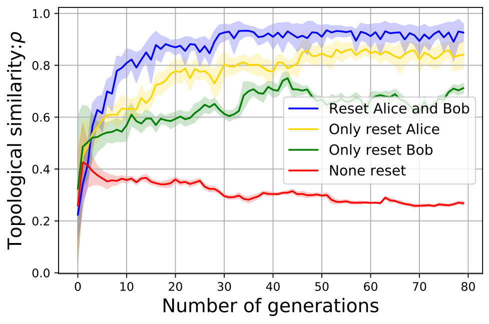
4 Experiments and Discussion

Due to the page limits, we only provide experimental results of the convergence behavior in terms of training accuracy, topological similarity, and zero-shot performance in this section. More detailed results, especially those about the posterior distribution of languages with different values of ρ, are provided in the appendix.

From Figure 2-(a), we can see that all three cases can almost perfectly play the game after some generations. The curve of the none-reset case will directly converge while the curves of the other two iterated learning cases will face accuracy degradation at the beginning of each generation. That is because, in the pre-train phase, Alice and Bob cannot perfectly learn from the data generated from the previous generation. Hence there should be a 're-match' training between them after reset. In Figure 2-(b), we can clearly see that the none-reset case has the lowest ρ while the iterated learning cases have higher ρ. Furthermore, we see the resetting of the Alice and Bob contributes separately to the performance hence the resetting-both case performs best among all. As resetting Alice plays a major role in iterated learning, the only-reset-Alice case convergence faster and have higher ρ comparing with the only-reset-Bob case. Such a trend can also be observed in Figure 3-(a), which records the zero-shot performance. In Figure 3-(b), each point represents a record of the zero-shot performance and the value of ρ at the same time, with the same experimental settings. From this figure, it is clear that zero-shot performance and topological similarity are linear correlated. The significance test result can also verify such a trend: the correlation ratio of them is 0.928, and the p-value of them is $3.8 * 10^{-104}$. In other words, the zero-shot performance and the value of ρ exhibit strong linear correlation under different algorithm settings. Hence it is reasonable to claim that the high-ρ emergent language generated by the agents using iterated learning indeed has higher compositionality comparing to that generated without iterated learning.

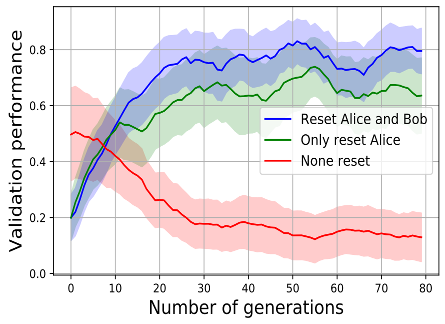


(a) Game performance

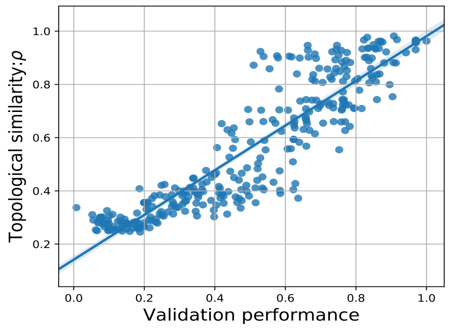


(b) Average ρ of emergent language

Figure 2: Convergence behavior under different algorithms in 80 generations.



(a) Zero-shot performance of different algorithms.



(b) Results of zero-shot performance and ρ.

Figure 3: Zero-shot performance and topological similarity.

References

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Brighton, H. (2002). Compositional syntax from cultural transmission. *Artificial life*, 8(1):25–54.

Brighton, H. and Kirby, S. (2006). Understanding linguistic evolution by visualizing the emergence of topographic mappings. *Artificial life*, 12(2):229–242.

Clark, H. H. (1996). *Using language*. Cambridge university press.

Cogswell, M., Lu, J., Lee, S., Parikh, D., and Batra, D. (2019). Emergence of compositional language with deep generational transmission. *arXiv preprint arXiv:1904.09067*.

Foerster, J., Assael, I. A., de Freitas, N., and Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145.

Havrylov, S. and Titov, I. (2017). Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *Advances in Neural Information Processing Systems*.

Kirby, S., Griffiths, T., and Smith, K. (2014). Iterated learning and the evolution of language. *Current opinion in neurobiology*, 28:108–114.

Kirby, S. and Hurford, J. R. (2002). The emergence of linguistic structure: An overview of the iterated learning model. In *Simulating the evolution of language*, pages 121–147. Springer.

Kirby, S., Tamariz, M., Cornish, H., and Smith, K. (2015). Compression and communication in the cultural evolution of linguistic structure. *Cognition*, 141:87–102.

Kottur, S., Moura, J., Lee, S., and Batra, D. (2017). 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.

Lazaridou, A., Hermann, K. M., Tuyls, K., and Clark, S. (2018). Emergence of linguistic communication from referential games with symbolic and pixel input. In *International Conference on Learning Representations*.

Lewis, D. (2008). *Convention: A philosophical study*. John Wiley & Sons.

Li, F. and Bowling, M. (2019). Ease-of-teaching and language structure from emergent communication. *arXiv preprint arXiv:1906.02403*.

Mordatch, I. and Abbeel, P. (2018). Emergence of grounded compositional language in multi-agent populations. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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

Appendix

A:PARAMETERS SETTING

Unless specifically stated, the experiments mentioned in this paper will select hyper-parameters following Table 1. The code for this paper will be released later.

Notation	Value	Description
N_a	2	Number of all attributes
N_v	8	Number of possible values for each attribute
N_L	2	Message length
$\|V\|$	8	Vocabulary size. (Here we have $V = \{abcdefgh\}$)
I	80, 100	Maximum number of generations
I_a	1200	Maximum pre-train **rounds** for Alice
I_b	200	Maximum pre-train **batches** for Bob
I_g	4000	Maximum game playing rounds
I_s	1000	Maximum rounds for sampling phase
N_h	128	Hidden layer size
N_b	64	Batch size
c	15	Number of candidates (including the target)
lr	$\geq 10^{-5}, \leq 10^{-3}$	Learning rate

Table 1: Value of hyper-parameters.

B: TOY EXAMPLE OF TOPOLOGICAL SIMILARITY

Group	Compsitional (8)	Holistic (16)	Other (232)
Language Examples	*blue box = aa*	*blue box = aa*	*blue box = aa*
	red box = ba	*red box = bb*	*red box = bb*
	blue circle = ab	*blue circle = ab*	*blue circle = aa*
	red circle = bb	*red circle = ba*	*red circle = bb*
ρ	1	0.5	0.1∼ 0.7

Table 2: Different groups of language and their topological similarity.

To better understand why topological similarity can represent the compositionality of one language, and also to provide some intuitions about what the mapping functions of languages with different values of ρ would be like, we provide a toy example in this appendix. In this example, we assume objects in $\mathcal{X}$ have two attributes ($N_a = 2$), each with two possible values ($N_v = 2$), then the object space contains four objects, say *(blue box), (blue circle), (red box)* and *(red circle)*. Suppose the message space is like (m_1, m_2), and each m_i have two possible values, i.e., $N_L = 2$ and $V \in \{a, b\}$, then the message space also contains four elements, say *aa, ab, ba* and *bb*. Any set of mappings from four distinct objects to four messages (messages can be the same) forms a language. We have 256 possible languages in this toy example, but only some of them can unambiously describe the four possible objects (there are 24 such languages in this example). Following the principles provided in (Kirby et al., 2015), we divide these 24 languages into two groups: compositional languages and holistic languages, and call the rest of 232 other languages, as illustrated in Table 2. The compositional languages, which are more similar to a human language, exhibit a clear structure when forming messages, while the holistic languages do not. In the compositional language in Table 2, m_1 and m_2 stands for the color and shape, respectively. We can hence use $S \to X + Y$, and X : *blue=a, red=b*; Y : *box=a, circle=b* to represent such a language. But in holistic or other languages, such a structure may not exist.

Note that the number of compositional languages are usually smaller than that of holistic languages. When the neural agent is randomly initialized, all possible mappings (i.e., languages) in the searching space follow a uniform distribution. Hence the initial probability of specific groups of languages can be calculated using the ratio of such language types to all possible language types. Using permutation and combination, we can calculate the numbers of unambiguous language, compositional language and holistic language as:

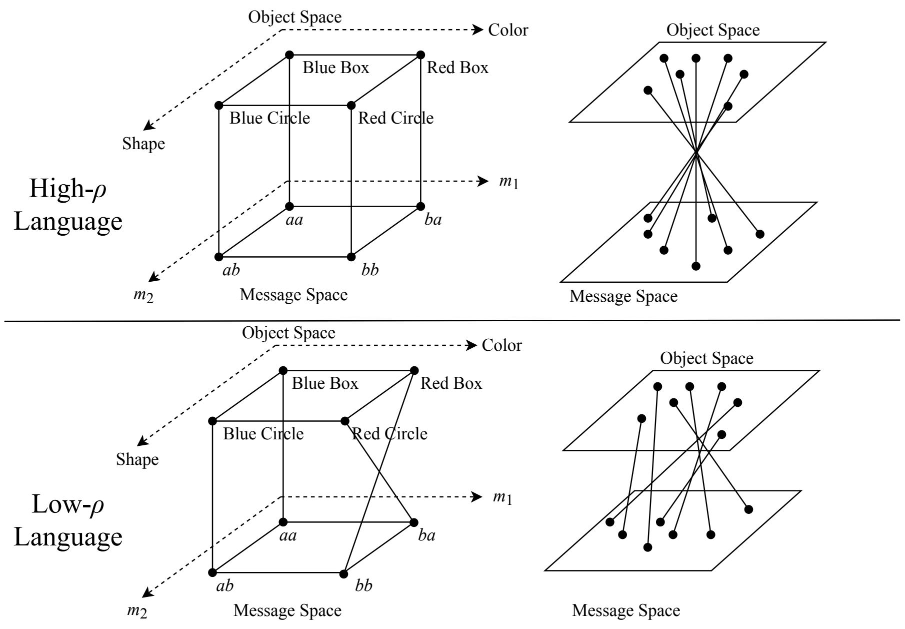


Figure 4: Examples of topological similarity of high-ρ languages and low-ρ languages.

$$\#\text{ unambiguous languages} = \frac{(|V|^{N_L})!}{(|V|^{N_L} - N_v^{N_a})!} \tag{4}$$

$$\#\text{ compositional languages} = N_a! \cdot \left(\frac{|V|!}{(|V| - N_v)!}\right)^{N_a} \tag{5}$$

$$\#\text{ holistic languages} = \#\text{ unambiguous languages} - \#\text{ compositional languages} \tag{6}$$

From the above equations, we can see that when N_L and $|V|$ increase, the gap between the number of compositional languages and that of holistic languages will become larger, which means that it is harder for us to select a compositional language randomly. That is why the expected topological similarity of the emergent language may increase when smaller N_L and $|V|$ are applied, as illustrated in (Lazaridou et al., 2018; Cogswell et al., 2019).

Besides the difference in the initial probability, another key difference between these two types of languages can be illustrated using topological similarity. As the language studied in this paper is defined as a mapping function from a meaning (i.e., the object) to a message, a compositional mapping must ensure that the meaning of a symbol is a function of the meaning of its parts. In other words, the compositional language is neighborhood related: nearby meanings tend to map to nearby signals, because nearby meanings usually share similar attributes and hence are likely to share similar message symbols (Brighton and Kirby, 2006).

Figure 4 provides a sketch of the topological similarity of high-ρ languages and low-ρ languages, from which we can get intuitions about the correlation between topological similarity and complexity of the corresponding mapping function. Generally, the mappings of high-ρ languages (e.g., compositional language) tend to have fewer inflection points and hence are more smoothing comparing with that of low-ρ languages. That is because high topological similarity will guarantee that any two close located objects will also be relatively close to each other after mapped to the message space. Thus for a language with $\rho = 1$, as illustrated in the figure, the mapping function of this language should be monotonic.

C: MORE EXPERIMENTAL RESULTS

Experiments of convergence behavior

We will provide some supplementary experimental results in this appendix to better explain our algorithm and analysis. We first provide the curve of the number of message types applied by Alice in each generation, as illustrated in Figure 5. As there are 64 different objects in $\mathcal{X}$ and 64 different message types in $\mathcal{M}$, we expect that all the message types are allocated to different objects. In other words, the message types applied by Alice should be close to 64. In this figure, we can see that the speaking agents in all the methods can apply at least 56 types of messages, which prove that the agents are not using the order of the target to play the game. Furthermore, after several generations, the resetting-both method performs slightly better than the others in terms of the message types, which means that the message type metric can also benefit from the iterated learning.

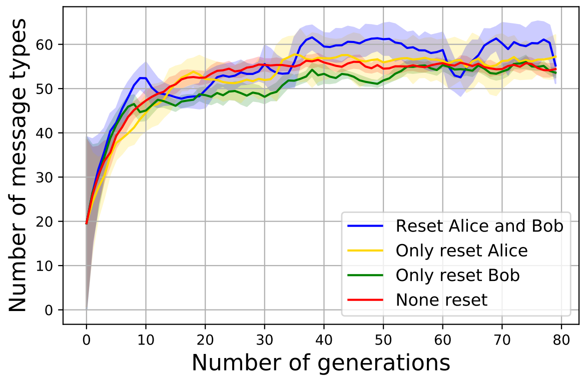


Figure 5: Message types of different settings.

We then provide two examples of the converged language (i.e., the language generated by Alice in the 79-th generation using argmax after softmax layer) using the none-reset method and the resetting-both method in Table 3 and 4, respectively. In these examples, both of the two languages can almost unambiguously represent all 64 different types of objects in $\mathcal{X}$, and hence they can help Alice and Bob to play the game successfully. However, it is obvious that the language generated using iterated learning has a clear structure: the first position of the message represents different colors, and the second position represents the shape. Such a structure is quite similar to what we humans do, i.e., combine an adjective and a noun to represent a complex concept.

Experiments of posterior probability distribution

To better illustrate the posterior probability of emergent languages with different values of ρ, we provide the 3D views of $P(\rho(\mathcal{L})|D_{i-1})$ in 80 generations in Figure 6. The heat-map provided in Figure 7 can be considered as the top views of these 3D illustrations. In these two figures, the x-axis and y-axis represent the index of generation and the topological similarity, and the z-axis represents the probability of languages with a specific value of ρ, in a specific generation. To make the figures more natural to read, we smooth the distribution of ρ in each generation using linear interpolation (Boyd and Vandenberghe, 2004).

Figure 8-(a) and (b) compare the posterior distributions at some typical generations, which can also be considered as the side views of the 3D illustration from the direction of x-axis. In these figures, we find that the initial distribution of ρ is not flat. That is because even the prior probability for each language is uniform, the amounts of languages with extremely high ρ and low ρ only occupy a small portion among all possible languages, as stated in (Brighton, 2002). Hence the initial probability of $\rho(\mathcal{L})$ is no longer uniform and has a bell shape which is similar to the Gaussian distribution. One new trend provided by these figures is that, in the none-reset case, the width of the curves in different generations do not change much, while in the resetting-both case, the width of the curves will gradually decrease (i.e., becomes more peaky). Such a trends means when iterated learning is applied, language tend to converge to some high-ρ types.

Figure 9-(a) and (b) track the ratio of languages with different values of ρ, which can also be considered as the side views of the 3D illustration from the direction of y-axis. In these figures, we divide all possible languages into five groups based on their topological similarity, i.e., languages with $\rho \leq 0.2$, $0.2 < \rho \leq 0.4$, $0.4 < \rho \leq 0.6$, $0.6 < \rho \leq 0.8$, and $0.8 < \rho$. We plot the ratio of these five different groups of languages at the end of each generation. From Figure 9-(a), we can see that the high-ρ language, which is represented by the bold curve, always occupy a small portion. The topological similarity of the dominant languages are $\rho < 0.4$. However, in the resetting-both case, as illustrated in Figure 9-(b), the portion of high-ρ language will increase significantly, which further verifies that the iterated learning can gradually make the high-ρ language dominate in posterior.

Experiments of zero-shot performance

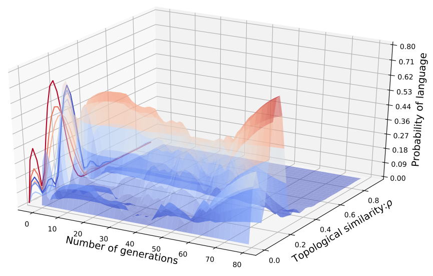
Besides the results provided in Figure 3, we further analyze the correlation between zero-shot performance and ρ under different settings. In Figure 10-(a), we separate all the data samples into two groups: with iterated learning and without iterated learning. In the significance test, the correlation ratio and p-value of the iterated learning case are 0.905 and $5.55 * 10^{-120}$, while for the no-iterated-learning case, the correlation ratio and p-value are 0.790 and $2.84 * 10^{-18}$. In other words, the slope of the regression line for the iterated learning case will be larger than that of no-iterated-learning case. However, for the iterated learning case with different hyper-parameters, the slopes of their regression lines are similar, but the y-intercept point of them are different, as illustrated in Figure 10-(b). The correlation ratio of these three cases (i.e., $I_a = 1200$, $I_a = 600$, and $I_a = 120$) are: 0.712, 0.700, and 0.401. The p-value of them are $9.55 * 10^{-10}$, $4.90 * 10^{-13}$ and $2.40 * 10^{-4}$. In summary, the zero-shot performance and the value of ρ exhibit strong linear correlation under different algorithm settings. Hence the idea of iterated learning has the potential to be extended to more general neural NLU systems.

	blue	green	cyan	brown	red	black	yellow	white
box	aa	fh	af	hh	cg	fc	ha	hf
circle	da	df	hb	db	fa	da	dh	fb
triangle	gc	ff	ge	gf	gg	fg	ge	he
square	ae	fb	be	bb	bg	fb	gb	ba
star	ad	fd	de	db	dg	fd	ce	hc
diamond	ac	dd	dc	db	dg	fd	dc	dd
pentagon	ad	fe	ef	bd	eg	fc	ee	ed
capsule	aa	dd	de	db	dg	gd	de	fh

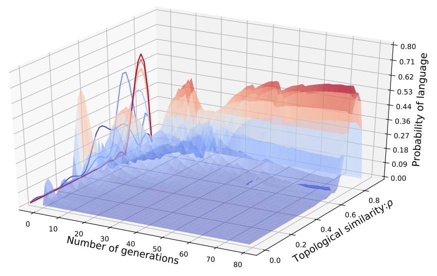
Table 3: Example of the converged language in none-reset case $\rho = 0.23$

	blue	green	cyan	brown	red	black	yellow	white
box	aa	ea	ba	ga	da	ca	ha	fa
circle	ab	eb	bb	gb	db	cb	hb	fb
triangle	ae	**eb**	be	ge	de	ce	he	fe
square	af	ef	bf	gf	df	cf	hf	ff
star	ac	ec	bc	gc	dc	cc	**dh**	fc
diamond	ad	ed	bd	gd	dd	cd	hd	fd
pentagon	ag	eg	bg	gg	dg	cg	hg	fg
capsule	ah	eh	bh	gh	**hc**	ch	hh	fh

Table 4: Example of the converged language in resetting-both case $\rho = 0.93$

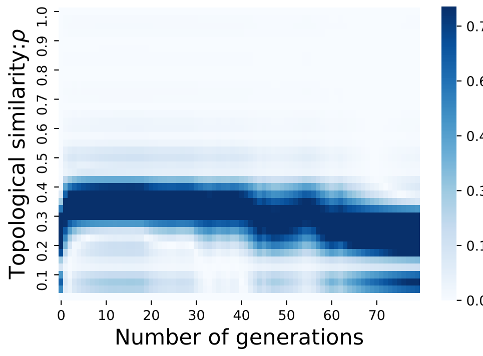


(a) None-reset case.

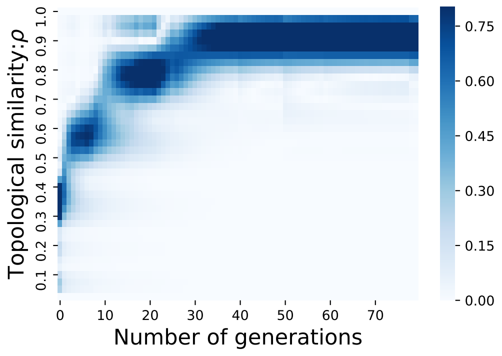


(b) Resetting-both case.

Figure 6: Language evolution of two cases in a 3D illustration.

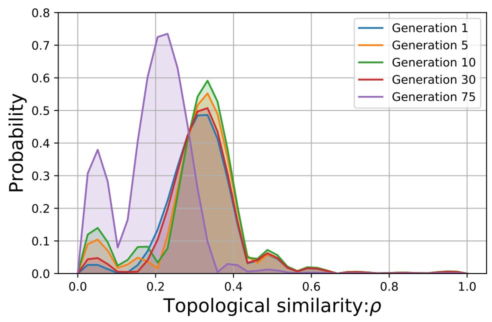


(a) The none-reset case.

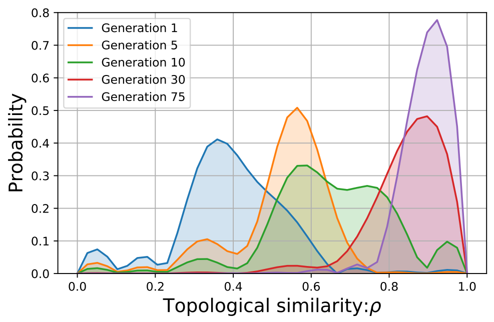


(b) The resetting-both case.

Figure 7: Distribution of $P(\rho(\mathcal{L})|D_{i-1})$ through 80 generations. Values of ρ are divided into ten groups. The distribution of ρ in each generation is smoothed using linear interpolation.

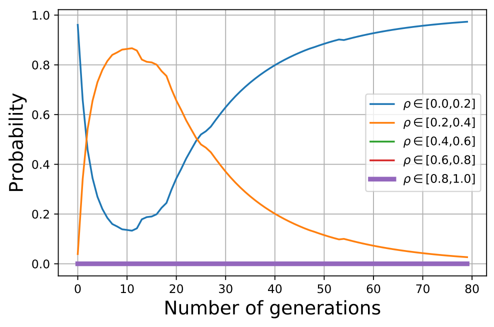


(a) None-reset case.

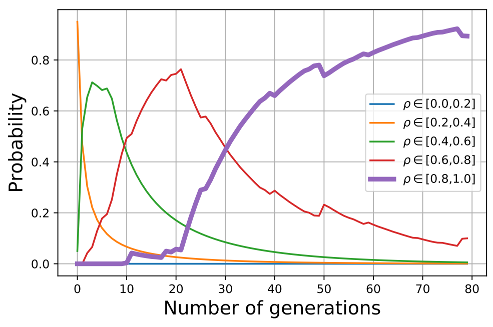


(b) Resetting-both case.

Figure 8: Distribution of $\rho(\mathcal{L})$ at different generations.

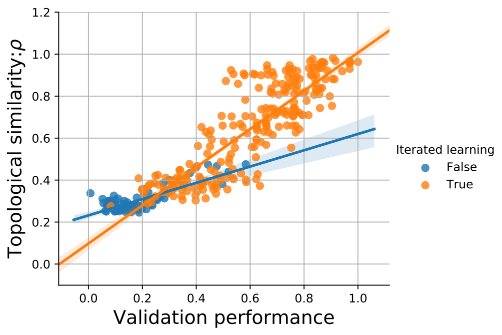


(a) None-reset case.

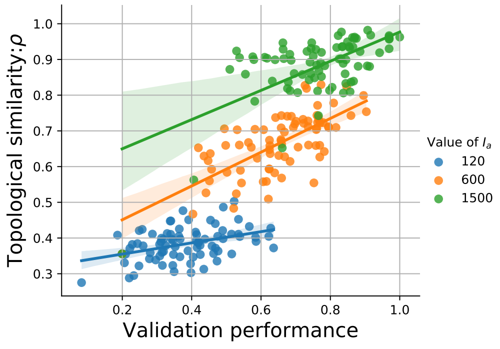


(b) Resetting-both case.

Figure 9: Evolution of language with different values of ρ.



(a) Whether iterated learning is applied.



(b) Different values of I_a.

Figure 10: Correlation between zero-shot performance and ρ under different settings.