模式分析与机器智能
工业和信息化部重点实验室
MIIT Key Laboratory of
Pattern Analysis & Machine Intelligence

ParNeC 模式识别与神经计算研究组
PAttern Recognition and NEural Computing

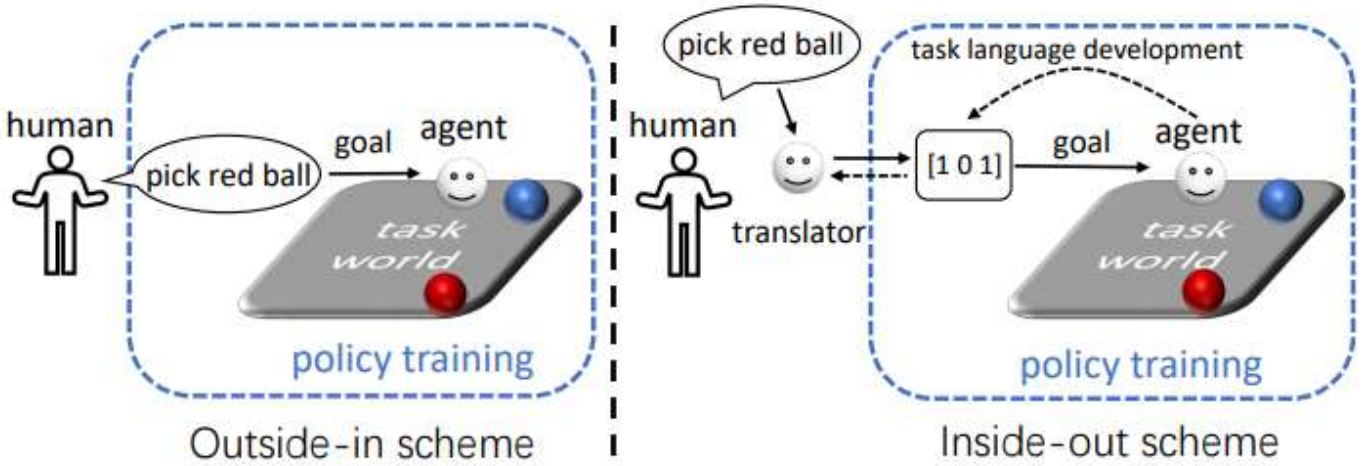
Natural Language-conditioned Reinforcement Learning with Inside-out Task Language Development and Translation

Jing-Cheng Pang [*1,2], Xin-Yu Yang [*1,2], Si-Hang Yang[1,2], and Yang Yu[†1,2]

[1]National Key Laboratory for Novel Software Technology, Nanjing University
[2]Polixir Technologies

NIPS 2024

Motivation



Set natural language as the task goal is not sufficient enough.

Coarse Experiments Settings

Definition 1 (**Task dataset**). *A task dataset* $\mathcal{D} = \{(s, s', L_N)_i\}$ *consists of multiple triplets. Each triplet contains a natural language sentence* L_N *and a task state pair* (s, s')*, where* L_N *describes state change from* s *to* s' *in natural language.*

To acquire the task dataset, we first train a policy that could move any specified ball to a specified position with PPO algorithms. Then, this policy will collect 100,000 state transitions, each corresponding to one random ball movement. Then, each state transition is assigned an NL description. We use Bert-base-uncased

Extract predicate representation named Task Language TL from (s, s') and map corresponding Natural Language NL to TL.

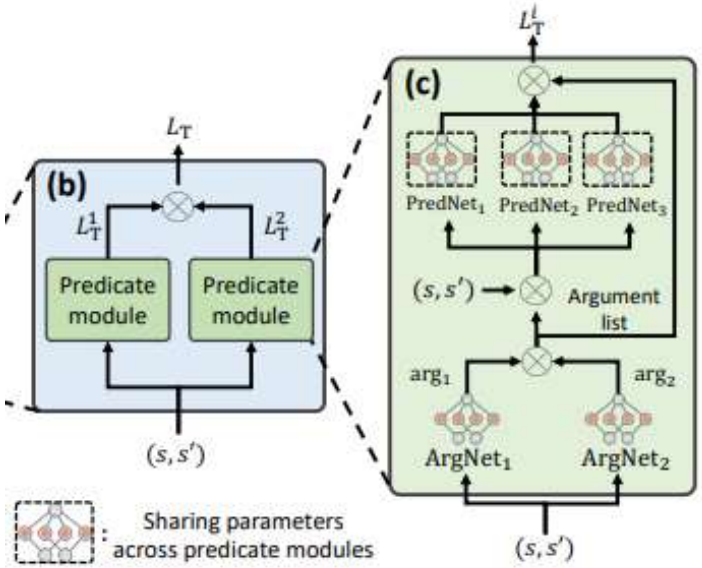
When Policy Training, Sample random NL as instructions to make agent learn to follow.

Predicate representation-Task Language

Compositional: blue circle and red square => blue square.

Interpretability: Pred(x, y) = True means some relations exist between x and y.

Figure 11: An illustration of the interpretability of predicate representation. For an anonymous predicate expression: `Pred(cat,grassland)=True`, we can guess that `Pred` represents [above].



predicate representation vector [1, 1, 0, 0, 0, 1, 0]

`Pred(a,b)`

Map Natural Language To Task Language

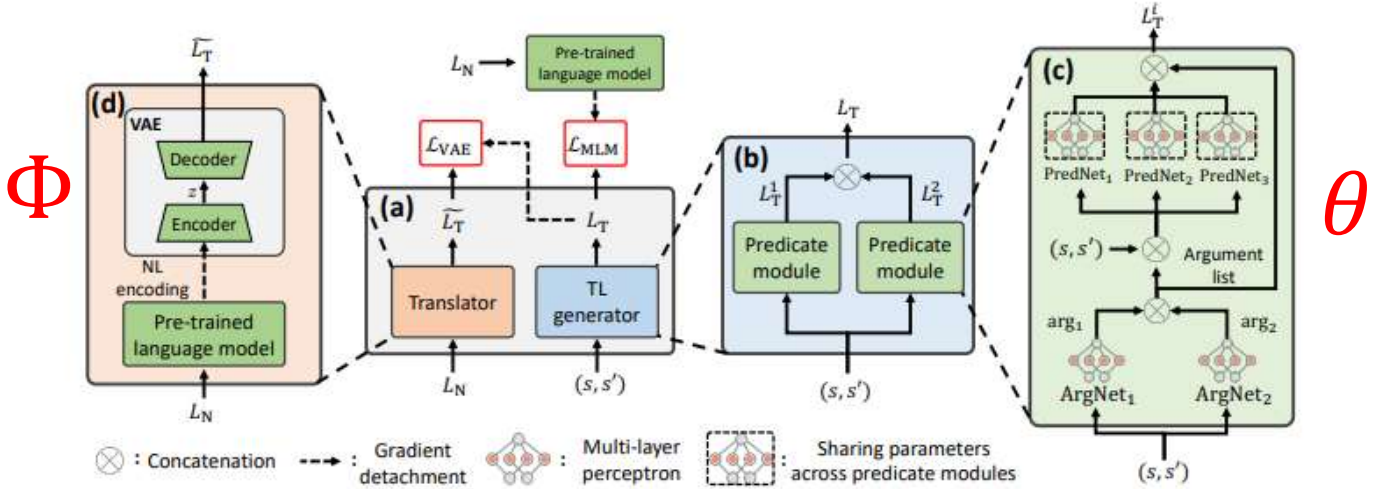


Figure 2: Overall training process of task language development and translation. **(a)** The overall training process. **(b)** Network architecture of the TL generator. **(c)** Architecture of one predicate module. **(d)** Network architecture of the translator. The number of predicate modules, arguments and predicate networks can be adjusted according to the task scale.

$$\mathcal{L}_{\mathrm{MLM}}(\theta) = \mathop{\mathbb{E}}_{(s,s',L_{\mathrm{N}})\sim\mathcal{D}} \left[-\sum_{i\in M} \log f(T_i | b(T_{\backslash M}), g_\theta(s,s')) \right], \tag{2}$$

where M denotes the set of the masked positions, $T_{\backslash M}$ denotes the masked version of L_{N}'s tokens, T_i is the i-th token, b is the Bert model, and f is a fully-connected network. Note that f is also optimized via gradient backpropagation. We omit the notion of its parameters for simplicity.

$$L_{\mathrm{VAE}}(\phi_1,\phi_2) = \mathop{\mathbb{E}}_{(s,s',L_{\mathrm{N}})\sim\mathcal{D}} \left[D_{\mathrm{KL}}(q_{\phi_1}(z|L_N), p(z)) - \mathop{\mathbb{E}}_{z\sim q_{\phi_1}(\cdot|L_{\mathrm{N}})} \left[\log p_{\phi_2}(L_{\mathrm{T}}|z) \right] \right], \tag{3}$$

where $L_{\mathrm{T}} = g_\theta(s,s')$, $z \sim q_{\phi_1}(\cdot|L_{\mathrm{N}})$ is the encoding generated by VAE encoder, and D_{KL} stands for KL-divergence.

Training Procedures

Algorithm 1 Training procedure of the TL generator.

Input: task dataset $\mathcal{D} = \{(s, s', L_{\mathrm{N}})_i\}$, pre-trained Bert model b.
Output: the optimized TL generator.

Initialize the TL generator g_θ.
while training not complete **do**
  Sample a batch of data from $\mathcal{D}$.
  Update θ to minimize the MLM loss (Eq.(2)).
end while
return the optimized TL generator g_θ.

Algorithm 2 Training procedure of the translator.

Input: task dataset $\mathcal{D} = \{(s, s', L_{\mathrm{N}})_i\}$, the optimized TL generator g_θ, and the pre-trained Bert model.
Output: the optimized TL generator.

Initialize the translator t_{ϕ_1,ϕ_2} with parameters ϕ_1 and ϕ_2.
while training not complete **do**
  Sample a batch of data $\{(s, s', L_{\mathrm{N}})_j\}$ from $\mathcal{D}$.
  // Compute the target task language which VAE aims to recover.
  Calculate the task language $L_{\mathrm{T}} = g_\theta(s, s')$.
  Update ϕ_1, ϕ_2 to minimize the VAE loss (Eq.(3)).
end while
return the optimized translator t_{ϕ_1,ϕ_2}.

Algorithm 3 Training procedure of the instruction-following policy.

Input: the optimized translator t_{ϕ_1,ϕ_2}.
Output: the optimized instruction-following policy.

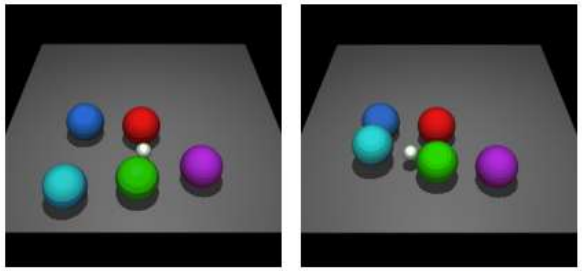
Initialize the policy function π, and the value function.
while training not complete **do**
  Sample a NL instruction L_{N} from the environment.
  Generate corresponding task language $\widetilde{L_{\mathrm{T}}} = t_{\phi_1,\phi_2}(L_{\mathrm{N}})$.
  // Collecting samples
  while episode not terminal **do**
    Observe current state s_t.
    Execute action $a_t \sim \pi(\cdot|s_t, \widetilde{L_{\mathrm{T}}})$, and receive a reward r_t from the environment.
  end while
  // Training
  Update the policy and value functions based on the samples collected from the environment.
end while
return the optimized policy π.

Experiments

- —(**Training set**)—
- Push the blue ball to the right of the green ball.
- Can you push the red ball to the right of the green ball?
- Can you help me push the red ball to the right of the green ball?
- Is the red ball right of the green ball?
- Is there any red ball right the green ball?
- The red ball moves to the right of the green ball.
- The red ball is being pushed to the right of the green ball.
- The red ball is pushed to the right of the green ball.
- The red ball was moved to the right of the green ball.
- —(**Testing set**)—
- Keep the red ball right of the green ball.
- Move the red ball to the right of the green ball.
- Can you move the red ball to the right of the green ball?
- Can you keep the red ball to the right of the green ball?
- Can you help me move the red ball to the right of the green ball?
- Can you help me keep the red ball to the right of the green ball?
- The red ball is being moved to the right of the green ball.
- The red ball is moved to the right of the green ball.
- The red ball was pushed to the right of the green ball.

We also consider a set of errors in the NL sentence:

- Missing a [the].
- Incorrect use of prepositions, e.g., using [*on front of*] to replace [*in front of*].
- Incorrect use of phrase, including *in behind of*, *in left of*, and *in right of*.
- Oversimplifying the expression, e.g., move red ball right green ball.



(a) Beginning of a task (b) Termination of a task

Figure 3: A visualization of CLEVR-Robot environment in our experiments. (a) In the beginning, one NL instruction is randomly sampled as *Can you move the cyan ball in front of the blue ball?* Then agent executes actions to complete the instruction. (b) The task terminates if achieving the goal or reaching the maximum timestep.

Accuracy

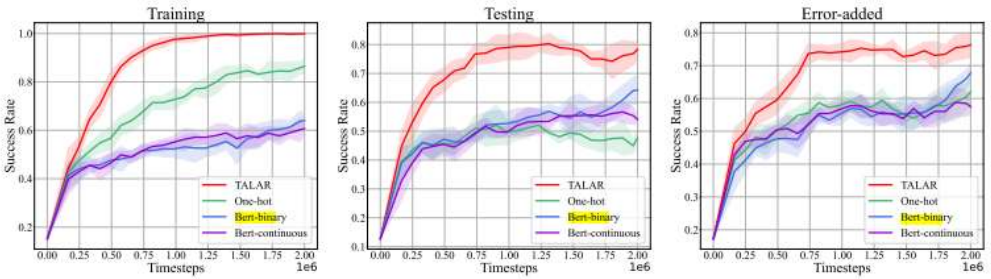


Figure 6: Training curves of different methods on three NL instruction datasets. The x-axis represents the total timesteps agent interacts with the environment, and the y-axis represents the success rate of completing instructions. The shaded area stands for the standard deviation over five random trials.

Method \ Dataset	Training	Testing	Error-added
TALAR	**99.9± 0.1**	**78.3 ± 3.1**	**76.3 ± 3.6**
One-hot	86.5 ± 3.0	47.6 ± 2.5	62.1 ± 4.1
Bert-binary	64.0 ± 2.5	64.2 ± 5.0	67.8 ± 2.9
Bert-continuous	60.7 ± 1.9	54.0 ± 1.3	57.5 ± 2.1

Bert-binary: Bert code to binary vector as TL.
Bert-continuous: Bert code to same size vector.
One hot: identify all task using a specific code.

Predicate Representation Interpretability

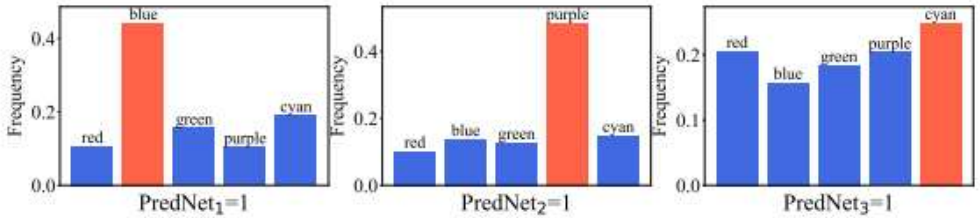


Figure 5: Frequency of five destination balls when a predicate network outputs a value of 1. Each bar stands for the frequency of the ball with a certain colour.

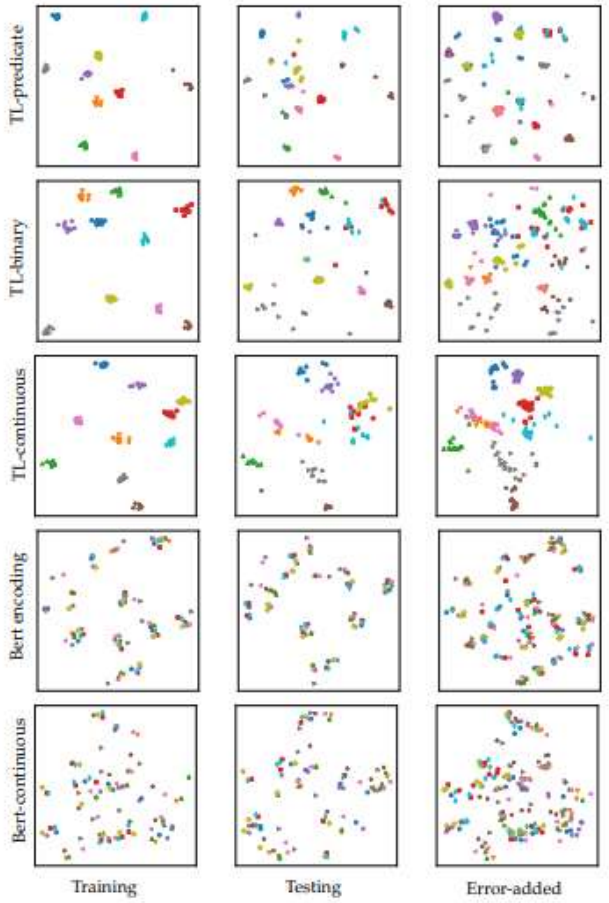


Figure 12: The t-SNE projection of different representations on three NL expressions datasets. Each row represents one kind of representation, and each column stands for one kind of NL expression. Points with the same marker encode nine different NL expressions that describe the same human instruction.

Ablation

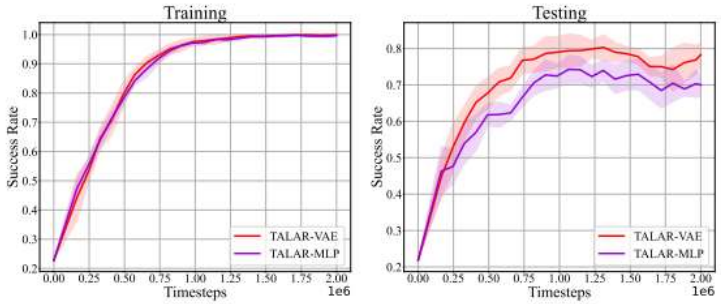
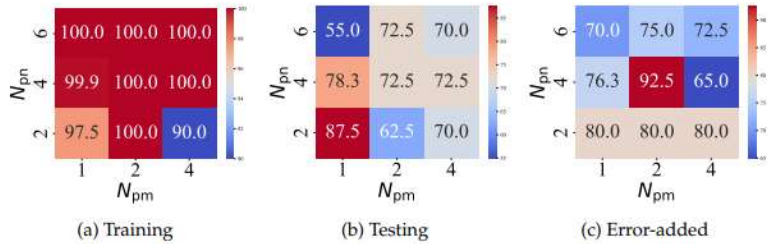


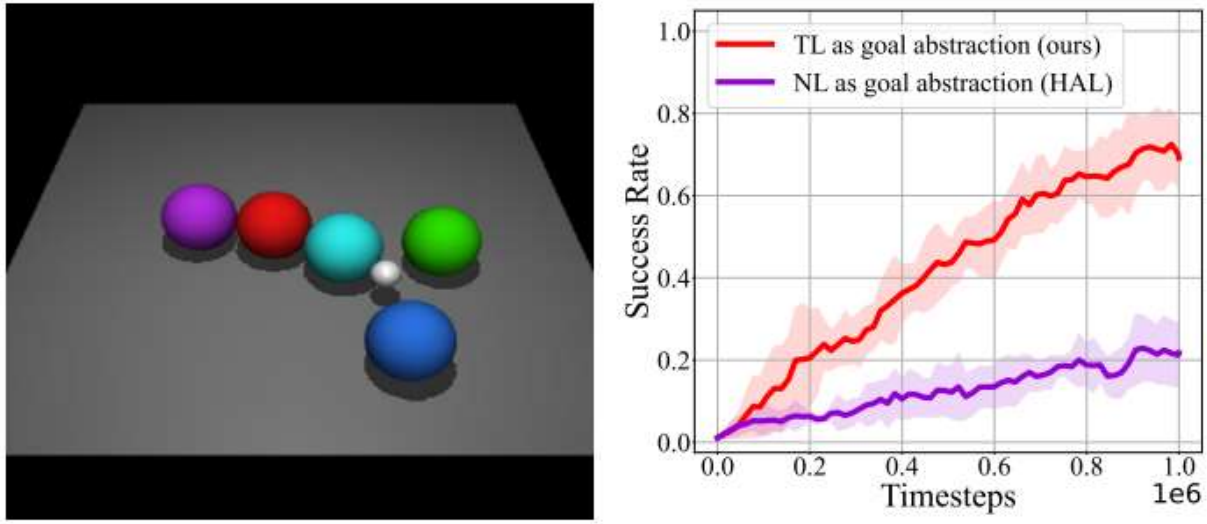
Figure 9: Training curves of IFP with different structures of the translator. See Appendix E.5 for the training curves on the error-added set.



(a) Training (b) Testing (c) Error-added

Figure 10: Ablation study on the number of predicate modules/networks. The values in the heat map represent the success rate of IFPs trained for 2M timesteps, with different parameter configurations of N_{pm} and N_{pn}.

instruction set (see Figure 10a). However, the experimental performance on the testing/error-added set is quite the opposite, as shown in Figure 10c when $N_{pm} = 4$ and $N_{pn} = 6$. This result could be attributed to the fact that, as the number of predicate modules and networks increases, the representation of TL becomes more complex (i.e., the vector size increases), making it more difficult for the policy to follow the TL. Besides, we also observe that, within a specific range of values ($N_{pm} \leq 2$ and $N_{pn} \leq 4$), larger N_{pn} and N_{pm} would bring a better performance. These results serve as a guide for selecting appropriate hyper-parameters. Due

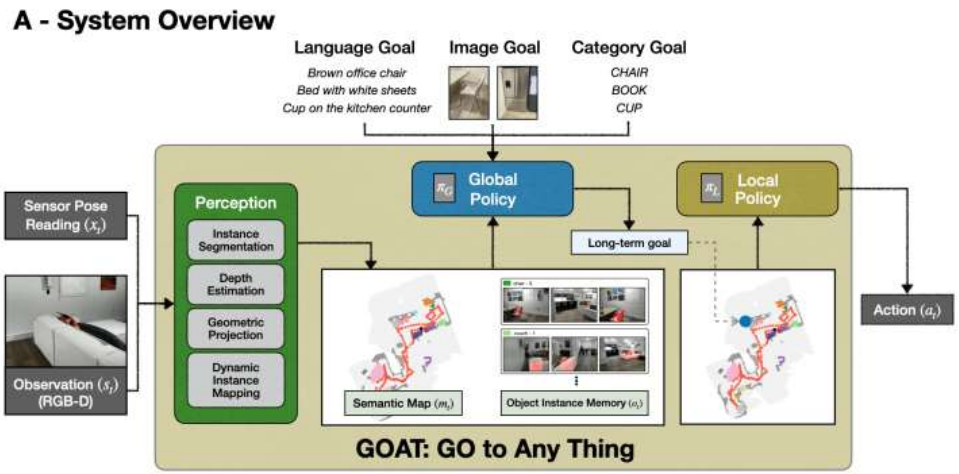


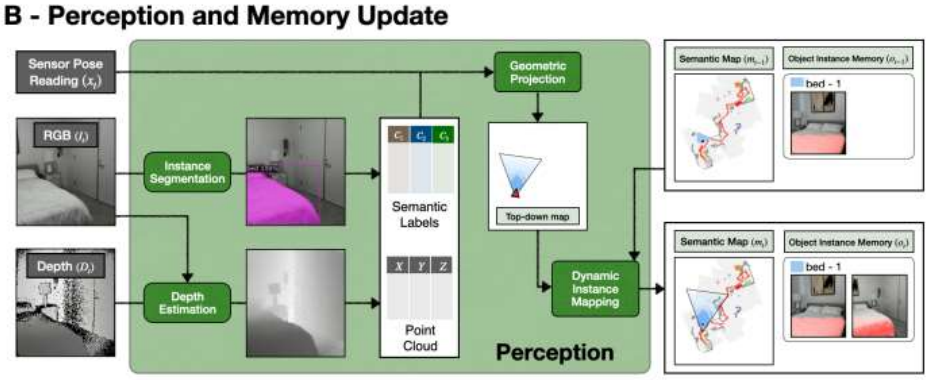
(a) An example of successful object arrangement.

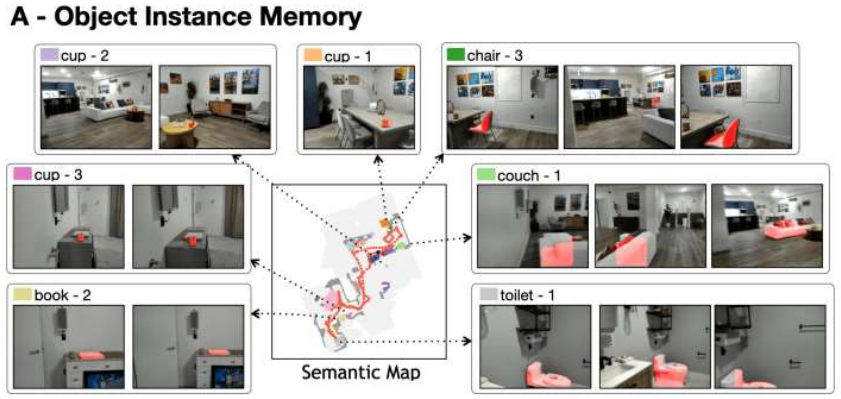
(b) Success rate curves during the training process.

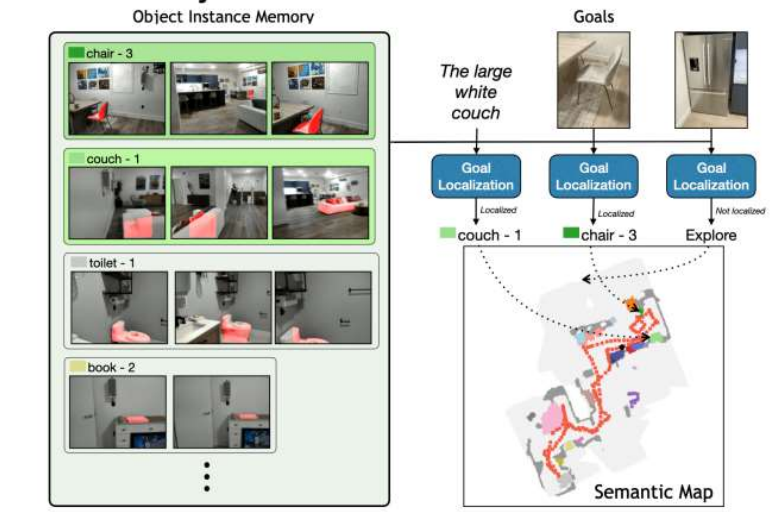
Figure 7: Experiments in a long-term object arrangement task. (a) A snapshot of successful object arrangement, which aims to arrange objects to satisfy all 10 human instructions at the same time. (b) Training curves of different goal abstractions in object arrangement.

Frontiers-GOTA









Thanks