Are you doing what I say?
On modalities alignment in ALFRED

Ting-Rui Chiang * and **Yi-Ting Yeh*** and **Ta-Chung Chi** and **Yau-Shian Wang**
Carnegie Mellon University
{tingruic,yitingye,tachungc,yaushiaw}@andrew.cmu.edu

Abstract

ALFRED is a recently proposed benchmark that requires a model to complete tasks in simulated house environments specified by instructions in natural language. We hypothesize that key to success is accurately aligning the text modality with visual inputs. Motivated by this, we inspect how well existing models can align these modalities using our proposed intrinsic metric, boundary adherence score (BAS). The results show the previous models are indeed failing to perform proper alignment. To address this issue, we introduce approaches aimed at improving model alignment and demonstrate how improved alignment, improves end task performance.

1 Introduction and Problem Definition

Recently several datasets have been proposed to benchmark machine's capability of following instructions in simulated real world environments (Anderson et al., 2018; Krantz et al., 2020; Wu et al., 2018; Hermann et al., 2020; Misra et al., 2018; Chen et al., 2019; Shridhar et al., 2020). To accomplish these tasks well, an agent needs to map the instruction given in natural language to actions while interacting with the environment simulator. ALFRED (Shridhar et al., 2020) is one of them that requires complex interaction. Each task in ALFRED involves a long sequence of instructions. To carry out the given task, an agent needs to navigate through the environment and interact with objects by generating pixel-level masks. It is thus very similar to real world scenarios.

Given this task, we ask a question: "*To what degree can a model align the literal instructions with its real world interactions?* We hypothesize this to be key to success. As human beings, when we are following a sequence of instructions, we know what the instruction we are working on, so we know what to do, and what to do next after finishing this instruction. For this reason, we conjecture such alignment is also necessary for a model.

Motivated by our hypothesis, we first investigate the existing models' capability of aligning the literal instructions and the visual inputs it perceives. To quantify this capability, we propose an intrinsic metric Boundary Adhere Score (BAS). It measures how frequently the model is focusing on the corresponding step-by-step instruction when predicting an action. We inspect the two publicly available baselines, the Seq2Seq model proposed in (Shridhar et al., 2020) and MOCA (Pratap Singh et al., 2020). The results indicate that both of the two have sub-optimal alignment.

Therefore, we propose methods to improve the alignments from two aspects. 1) We propose a novel neural program counter model, which explicitly keeps track of the instruction that is being executed. 2) We propose a new auxiliary loss L_{pc} that provides stronger learning signal for learning the alignment.

Our preliminary results show that the program counter module along with the auxiliary loss are promising. It outperforms the original state-of-the-art model. We also apply our BAS metric on the MOCA model with our proposed program counter and auxiliary loss. The results show that they can indeed improve the alignment. It is aligned with our hypothesis that alignment is important.

To sum up, our contribution is three-fold: 1) We propose an intrinsic metric that quantifies the alignment. 2) Our analysis indicates that the previous models do not align the modalities very well. 3) Based on our analysis we propose improvement over the current state-of-the-art model, leading to a promising future research direction.

*Chiang and Yeh have equal contributions

2 Related Work

2.1 Vision and Language Navigation Tasks

Vision and language navigation tasks bear a strong resemblance to ALFRED, as they require an agent to achieve a goal by following instructions in a simulated environment.

R2R Stanford large-scale 3D Indoor Spaces (Armeni et al., 2016) and Room2Room (R2R) (Anderson et al., 2018) are two benchmarks for Vision-and-Language Navigation (VLN). In R2R, the whole system is built upon the Matterport-3D simulator, which encodes indoor environments as graphs where nodes are navigable waypoints and edges represent the feasibility to go from one waypoint to another.

VLN-CE Krantz et al.; Wu et al. further proposes Vision-and-Language Navigation in Continuous Environments (VLN-CE) which doesn't provide pre-defined navigation graphs to an agent. An agent needs to navigate via egocentric visual inputs as opposed to panoramic viewpoints.

StreetNav Other than indoor environments, StreetNav (Hermann et al., 2020) additionally provides thumbnails to an agent to navigate in Google Street View environment.

LANI Misra et al. proposes LANI where an agent navigates between landmarks, and CHAI where an agent navigates and manipulates objects in the house environment.

DeepMind Lab DeepMind Lab environment (Beattie et al., 2016) provides a set of first-person 3D mazes for an agent to navigate, and Mirowski et al. experiments the effect of different auxiliary losses on the mazes.

Touchdown In Touchdown dataset (Chen et al., 2019), in addition to reaching certain locations the agent also needs to use the spatial description to locate the mascot.

Unlike other navigation tasks, IQA (Gordon et al., 2018) provides agents with questions instead of the instructions. Agents need to explore the interactive environment to get necessary information to answer questions. While aforementioned datasets address different aspects of vision and language grounded tasks, ALFRED requires an agent to perform more complicated actions such as navigating via egocentric visual inputs and meanwhile interact with objects by predicting interaction masks.

2.2 Existing Techniques

Fried et al. uses a speaker model to generate instructions from simulated trajectories. Tan et al. applies the environmental dropout mask to remove some objects from the scenes.

Storks et al. proposes an approach that integrates an object detection model with a model that predicts the direction of the goal. Corona et al. uses separate modules to predict actions for different types of instructions. This leads to improved performance on unseen scenes and tasks. Pratap Singh et al. argues that the information for predicting object masks and actions is different (i.e. general instr v.s. step-by-step). By separating the model into object mask prediction and action prediction branches, they achieved the second best on the leaderboard.

Ma et al. proposes progress estimation that encourages the agent to utilize the instruction sequentially with attention. Relying on the estimated progress, Ma et al. proposes a regret module that decides whether or not to roll back, and use a progress marker for action selection. Hu et al. shows that visual features are not well utilized in unseen scenarios. They also propose to use Faster-R-CNN (Ren et al., 2015) to construct visual features. It performs better in unseen scenes.

There are also techniques for environments other than ALFRED and Room2Room. Gupta et al. proposes Cognitive Mapper and Planner, which learns the world map information with a differentiable spatial memory to select actions for an input goal. Blukis et al. grounds visual objects with a database for few-shot adaptation in a quadcopter simulator environment. Andreas and Klein formulates the instruction to action task as a structure alignment problem using conditional random field (CRF). Finally, there are some studies that formulate the task as an instruction parsing task. Karamcheti et al. designs a pipeline that parses instructions into primitive actions. Artzi and Zettlemoyer parses instructions with a combinatory categorial grammars (CCG) parser learned with a weakly supervised algorithm.

3 Background

3.1 Task Formulation

The ALFRED task (Shridhar et al., 2020) aims to learn a mapping from natural language instructions and egocentric vision to sequences of actions for household tasks. An agent needs to carry out

tasks described in given instructions by interacting with an environment. For each trajectory, at the beginning, a high-level instruction and some step-by-step instructions in natural language are given to the agent. At each time step, the agent has access to the visual observation from its first person point of view. The agent needs to output actions according to the instructions and the visual input. Possible actions include five navigation actions and seven actions that interact with objects in the environment. To interact with an object in the environment, for example, pick up an object, the model needs to generate a pixel-level mask that selects an object.

3.2 Dataset

The ALFRED training dataset consists of trajectories in different environments generated by expert demonstration, each of which is annotated with the corresponding instructions. We include the statistics about the dataset in Table 1. The dataset also provides the time span each step-by-step instruction corresponds to. We will use this additional information in our following analysis and proposed approaches.

3.3 Metrics

We consider two existing metrics used in ALFRED (Shridhar et al., 2020): 1) **Task Success**: It is defined as 1 if the task goal-conditions are all met, and 0 otherwise. 2) **Goal-Condition Success**: The task success is 1 only if all goal-conditions are met, which might be too challenging. To better justify the performance, we report the percentage of completed goal-conditions. In addition, as shorter trajectories are more efficient and favorable, we also consider the path weighted version of the above two metrics. Concretely, the weight to be multiplied with is calculated as $\frac{L^*}{\max(L^*,\hat{L})}$, where $\hat{L}$ is the number of actions taken by the expert, and L^* is the number of actions taken by the model.

3.4 Baseline Models

We choose two publicly available models as our baseline: (1) The **Seq2Seq** model proposed in the original ALFRED paper (Shridhar et al., 2020). Given visual observation, instructions and a goal as inputs, the Seq2Seq model is trained to predict the action sequence with imitation learning. (2) **MOCA** (Pratap Singh et al., 2020). It separates the model into a visual perception module predicting object masks and an action policy module predicting actions. Since these two predictions require different information, they argued that separating the model into two branches improves the overall performance, achieving the second best performance on the leaderboard when the time we do this work.

Both of the two models use two auxiliary losses.

1. A progress monitor (PM) loss: At each time step t, the model predict a progress $\hat{p}_t$. The PM loss is a l_2 loss between $\hat{p}_t$ and p_t, where $p_t = t/T$ is the progress in terms of the total number of steps T.

2. A sub-goal (sg) loss: At each time step t, the model predict the ratio of completed sub-goals $\hat{c}_t$. The PM loss is a l_2 loss between $\hat{c}_t$ and c_t, where $c_t = c/C$ is the number of completed sub-goals c over the number of sub-goals in this instance C.

4 Alignment Analysis

Motivation Intuitively, better alignment between modalities (i.e. visual representations and instructions) should lead to improved performance. However, previous work (Hu et al., 2019) discovers that models might achieve better performance with the removal of certain modalities, let alone the alignment between them. While ALFRED (Shridhar et al., 2020) justifies the necessity of all modalities for task success, the alignment between them is still not measured explicitly. Overall, two questions remain unanswered: 1) does the model learn alignment between modalities? 2) If yes, does better alignment lead to task performance improvement? We aim at designing metrics that help us answer the two questions.

Alignment Definition We are given two input modalities: a sequence of images, denoted by $\bar{V} = \{v_i\}_{i=1}^{T}$, and a sequence of words of step-by-step instructions , denoted by $\bar{S} = \{s_j\}_{j=1}^{L_s}$, where L_s is the word number of instructions and s_j is a word in instructions. The ground truth alignment α is defined to be a surjective function $f : \bar{V} \rightarrow \bar{S}$, which is given in the Alfred dataset.

Approaches We propose two approaches to identify model alignment f_M between $\bar{V}$ and $\bar{S}$ in a model $\mathcal{M}$. In the first method, we assume the existence of an attention mechanism between $\bar{V}$ and $\bar{S}$. Let the attention score from v_t to the kth token of

	Train	Valid-Seen	Valid-Unseen
# of Step-by-Step Instruction	6.72 (2.49)	6.79 (2.72)	6.26 (2.33)
# of word in Step-by-Step Instruction	12.30 (5.93)	12.13 (6.00)	12.59 (6.36)
# of word in High-level Instruction	10.06 (3.05)	10.11 (3.17)	10.05 (2.89)

Table 1: Statistics of the instructions.

	Train		Seen		Unseen	
	Attention	Gradient	Attention	Gradient	Attention	Gradient
random	.290 (.108)		.294 (.115)		.328 (.119)	
Seq2Seq (2020)	.590 (.155)	.594 (.158)	.589 (.153)	.593 (.156)	.354 (.182)	.363 (0=.181)
- p.m. loss	.575 (.147)	.580 (.149)	.567 (.147)	.571 (.150)	.368 (.191)	.372 (.188)
- subgoal loss	.541 (.146)	.554 (.149)	.544 (.149)	.554 (.149)	.369 (.177)	.382 (.178)
- both	.562 (.152)	.567 (.155)	.551 (.151)	.557 (.155)	.337 (.181)	.342 (.179)
MOCA (2020)	.443 (.207)	.382 (.180)	.450 (.208)	.384 (.177)	.436 (.202)	.348 (.165)

Table 2: Boundary Adherence Score of baselines. A higher score means less violation of alignment. "Random" is the expected score when the attention score or random score distributes uniformly over the instructions. The metrics on the training set is computed over 820 trajectories randomly sampled from the training data.

the jth instruction be $\alpha_{t,j}^{(k)}$. We define

$$f_M(v_t) = \arg\max_j \max_k \alpha_{t,j}^{(k)}. \quad (1)$$

Another way to identify f_M is to inspect the gradient norm of an input instruction. At each time step t, an ALFRED model predicts an action a_t according to the video inputs the model has observed $\{v_i\}_{i=1}t$. We calculate the gradient norm of the kth token in the jth instruction $s_j^{(k)}$ as:

$$g_{t,j}^{(k)} = \left\| \frac{\partial \mathcal{L}(a_t)}{\partial \mathrm{Emb}(s_j^{(k)})} \right\|_2, \quad (2)$$

where $\mathcal{L}(a_t)$ is the loss at time step t. A greater value of gradient norm implies the word contributes more to the action prediction. With g, we can define

$$f_M(v_t) = \arg\max_j \max_k g_{t,j}^{(k)}. \quad (3)$$

Note that the action $\{a_t\}$ serves as a proxy of V here due to the natural mapping between actions and images.

Soft alignment scores such as attention scores or gradient norm can always be transformed to hard alignment function f_M by greedy/max selection.

Boundary Adherence Score Finally, we define the *Boundary Adherence Score* (BAS) B as:

$$B = \frac{1}{L_s} \sum_{i=1}^{L_s} \mathbb{1}[f(v_i) = f_M(v_i)] \quad (4)$$

It is the frequency that the model's alignment follows the ground truth alignment. Obviously, higher B indicates better f_M.

BAS of the Baseline Models Table 2 shows the BAS of our two baseline models Seq2Seq and MOCA. Both the Seq2Seq and MOCA have higher alignment scores than the expected score of random alignment. This implies that the Seq2Seq model and MOCA are able to align the two modalities to some extent, Interestingly, MOCA with worse BAS in fact has a far better evaluation result than Seq2Seq. It shows the alignment ability of models might not be very indicative to the final results. To verify our motivation that the progress monitor auxiliary loss and the subgoal auxiliary loss are not sufficient for the learning of alignment, we also apply the intrinsic metrics on the Seq2Seq models trained without using auxiliary losses. We can see that not using the auxiliary losses affect the alignment metrics by less than 0.04. It indicates that the auxiliary loss is not very effective to encourage the learning of the alignment.

5 Models

5.1 Proposed Approaches

Based on the analysis of the intrinsic metric, we have found the models fail to align the modalities well. We also showed adding the progress monitor loss doesn't improve the learning of the alignment. We conjecture that this maybe be due to two issues 1) The current model architecture cannot utilize

the alignment well. 2) Neither progress monitor nor action prediction provides sufficient signal for learning the alignment.

5.1.1 Neural Program Counter

To address the first issue, we propose using a *neural program counter* (PC) that adds inductive bias forcing the model to use instructions sequentially. It is an analogy to the program counter in a CPU, where a program counter stores the index of the machine code to be executed. When it is increased by 1, the CPU will execute the next machine code. Here we treat each low-level step-by-step instruction as an atomic command, and then we define a *soft* neural program counter. Its value is relaxed to the continuous real number $\mathbb{R}^+$. Specifically, let the value of the neural program counter at the time step t be $c^{(t)}$. The value is initialized as 0:

$$c^{(0)} = 0. \tag{5}$$

At each action decoding step t, the model can decide whether or not to increase it:

$$c^{(t+1)} = c^{(t)} + \sigma(f_c(h^{(t)}), \tag{6}$$

where f_c is a linear layer, σ is the sigmoid function, and $h^{(t)}$ is the decoder hidden state at a time step t.

Given a sequence of words of step-by-step instructions $\bar{S} = \{s_j\}_{j=1}^{L_s}$, we use p_j^{instr} to denote the index of step-by-step instructions which the word s_j is in. For example, if the word s_j belongs to the third instruction, then p_j^{instr} is 2 since the index starts from 0. In both Seq2Seq baseline model and MOCA model, the decoder computes attention weights $a^{(t)}$ over input instruction words $\bar{S}$ at each decoding step. To force the model to focus on an specific instruction at each decoding step, we use $c^{(t)}$ to construct an attention mask $m^{(t)} \in \mathbb{R}^{L_s}$ on $a^{(t)}$:

$$m_j^{(t)} = \exp\left\{-\lambda \left| p_j^{instr} - c^{(t)} \right|\right\}, \tag{7}$$

where $m_j^{(t)}$ is the attention mask for the word s_j, and λ is a parameter to learn whose minimum value is 0. The λ controls the strictness of the mask. When $\lambda \to \infty$, $m_j^{(t)}$ becomes a hard 0-1 mask. On the contrary, when $\lambda \to 0$, $m_j^{(t)}$ is a mask whose all values are equal to 1. We expect the model can automatically adjust λ to make the learning process easier.

The mask $m_i^{(t)}$ is then applied to the original attention weights $a^{(t)}$ to form the new attention weights:

$$\tilde{a}^{(t)} = a^{(t)} \odot m^{(t)} \tag{8}$$

where $\odot$ denotes the element-wise product. The model then uses these new attention weights $\tilde{a}^{(t)}$ to compute the attention output as the original model predicts the action at this step.

5.1.2 Auxiliary Loss L_{pc}

However, having a program counter itself may not be able to address the second issue mentioned in Section 5.1. The training signal might not be sufficient for the model to learn a proper $c^{(t)}$, which would cause the masked attention $\tilde{a}^{(t)}$ to be very noisy. Therefore, we designed an auxiliary loss L_{pc} to make the learning of the proposed program counter easier. Since the training data provides that ground truth mapping between each action and the corresponding step-by-step instruction, we can compute the oracle program counter value $\bar{c}^{(t)}$ at each time step. The loss L_{pc} is the Mean Squared Error (MSE) loss between the oracle $\bar{c}^{(t)}$ and the predicted $c^{(t)}$.

$$L_{pc} = \frac{1}{T}\sum_{t=1}^{T}(\bar{c}^{(t)} - c^{(t)})^2 \tag{9}$$

where T the number of time steps in one sample.

5.1.3 Fine-grain Program Counter

We have also observed that there are long instructions which can be further split into multiple shorter sequences. Since generating attention masks over more fine-grained instructions might allow the model to learn better alignments, we propose to split the long instructions with punctuation and the word "and". For example, the instruction "Turn to the right and move towards the range, then turn to the right ..." will be split into three short instructions "Turn to the right", "move towards the range", and "then turn to the right ...". We name the program counter trained on these fine-grained instructions *Fine-grained PC*.

When the fine-grained PC is used, the context of a shorter instruction may become important. The agent may need to look ahead one instruction in order to know whether it has completed this instruction. It motivates us to propose to modify Equation 7 a bit:

$$\begin{aligned} m_j^{(t)} = \exp\{ &-\lambda \mathrm{ELU}\left(\left| p_j^{instr} - c^{(t)} \right| - \tau\right) \\ &+ \lambda \mathrm{ELU}(-\tau)\}. \end{aligned} \tag{10}$$

Methods	Valid Seen Task	Valid Seen G-C	Valid Unseen Task	Valid Unseen G-C	Test Seen Task	Test Seen G-C	Test Unseen Task	Test Unseen G-C
No Language	0.0 (0.0)	5.9 (3.4)	0.0 (0.0)	6.5 (4.7)	0.2 (0.0)	5.0 (3.2)	0.2 (0.0)	6.6 (4.0)
No Vision	0.0 (0.0)	5.7 (4.7)	0.0 (0.0)	6.8 (6.0)	0.0 (0.0)	3.9 (3.2)	0.2 (0.1)	6.6 (4.6)
Goal-Only	0.1 (0.0)	6.5 (4.3)	0.0 (0.0)	6.8 (5.0)	0.1 (0.1)	5.0 (3.7)	0.2 (0.0)	6.9 (4.4)
Instruction-Only	2.3 (1.1)	9.4 (6.1)	0.0 (0.0)	7.0 (4.9)	2.7 (1.4)	8.2 (5.5)	0.5 (0.2)	7.2 (4.6)
Shridhar et al. (2020)	3.7 (2.1)	10.0 (7.0)	0.0 (0.0)	6.9 (5.1)	4.0 (2.0)	9.4 (6.3)	0.4 (0.1)	7.0 (4.3)
Storks et al. (2021)	1.4 (0.0)	-	0.0 (0.0)	-	-	-	-	-
Okatani and Nguyen (2020)	-	-	-	-	12.4 (8.2)	20.7 (18.8)	4.5 (2.2)	12.3 (9.4)
Pratap Singh et al. (2020)	19.2 (13.6)	28.5 (22.3)	3.8 (2.0)	13.4 (8.3)	22.1 (15.1)	28.3 (22.1)	5.3 (2.7)	14.3 (10.0)
MOCA + Oracle PC	6.2 (4.3)	14.6 (12.5)	0.5 (0.2)	8.3 (7.2)	-	-	-	-
MOCA + PC	16.6 (12.1)	25.7 (20.6)	1.7 (0.8)	11.7 (7.9)	-	-	-	-
MOCA + PC + L_{pc}	**19.5 (14.7)**	**28.9 (24.0)**	3.9 **(2.3)**	13.3 (8.9)	-	-	-	-
MOCA + F.G. PC + L_{pc}	17.7 (12.1)	25.8 (20.6)	**4.1** (2.2)	**14.2 (9.1)**	-	-	-	-
MOCA + F.G. PC + L_{pc} + τ	15.7 (10.1)	24.0 (14.9)	3.4 (1.8)	14.0 (8.6)	-	-	-	-
Human	-	-	-	-	-	-	91.0 (85.8)	94.5 (87.6)

Table 3: Task and Goal Condition Success Rate. For each metric, the corresponding path weighted metrics are given in parentheses. All values are percentages.

	Train Attention	Train Gradient	Seen Attention	Seen Gradient	Unseen Attention	Unseen Gradient
random	.290 (.108)		.294 (.115)		.328 (.119)	
Seq2Seq (2020)	.590 (.155)	.594 (.158)	.589 (.153)	.593 (.156)	.354 (.182)	.363 (.181)
MOCA (2020)	.443(.207)	.382 (.180)	.450 (.208)	.384 (.177)	.436 (.202)	.348 (.165)
MOCA + Oracle PC	.990 (.033)	.732 (.135)	.989 (.037)	.736 (.136)	.990 (.040)	.718 (.140)
MOCA + PC	.448 (.135)	.364 (.143)	.429 (.133)	.345 (.144)	.424 (.148)	.336 (.144)
MOCA + PC + L_{pc}	.813 (.121)	.735 (.147)	.777 (.139)	.705 (.155)	.724 (.165)	.646 (.166)
MOCA + F.G. PC + L_{pc}	.566 (.147)	.559 (.155)	.556 (.156)	.550 (.165)	.501 (.147)	.492 (.151)
MOCA + F.G. PC + L_{pc} τ	.595 (.145)	.545 (.159)	.579 (.158)	.537 (.162)	.534 (.149)	.497 (.154)

Table 4: Boundary adherence score. A higher score means less violation of alignment. “Random" is the expected score when the attention score or random score is distributed uniformly over the instructions. The metrics on the training set is computed over 820 trajectories randomly sampled from the training data.

Given $c^{(t)}$, $m_j^{(t)} = 1$ if $p_j^{instr} = c^{(t)}$, which is the same as in Equation 8. However, when $c^{(t)}$ differs p_j^{instr} by less than τ, the attention weights over the jth tokens $m_j^{(t)}$ in the instruction will only be less than 1 by a little. As a result, the model will be able to attend tokens in the instructions ranging from $c - \tau$ to $c + \tau$. We use $\tau = 1$ in the following experiments, and name this approach *Fine-grained PC* τ.

6 Experimental Setup

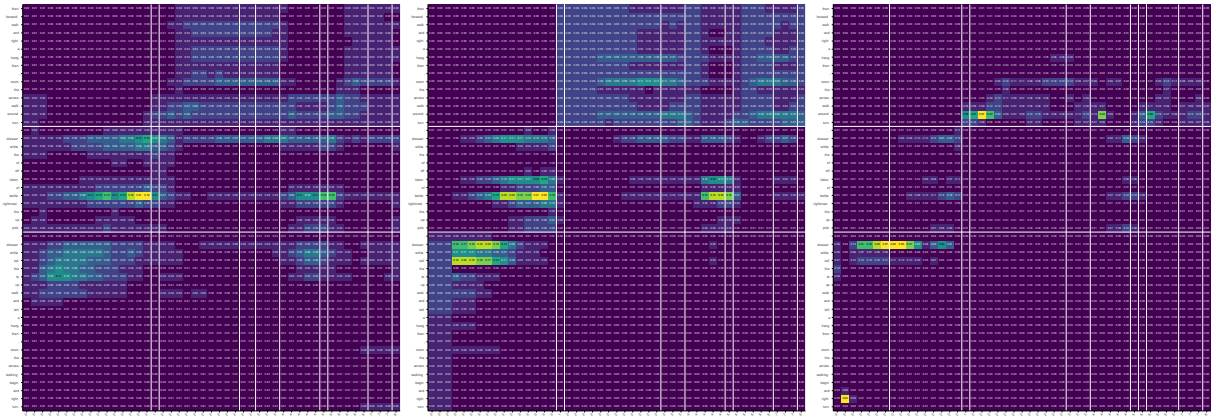
We conduct our experiments on the ALFRED dataset following the setting in the (Pratap Singh et al., 2020). Due to the constraint of computing resources, we train our models with fewer update steps. When the auxiliary loss L_{pc} is used, we weight it by 0.2.

7 Results and Discussion

7.1 Neural Program Counter

The results are shown in Table 3. When the program counter and L_{pc} are used, our model is able to outperform both the reported and reproduced MOCA models on seen and unseen environments. The proposed model has larger performance gain on the path weighted metrics, which indicates the model with PC and L_{pc} does less redundant actions. It shows the superiority of the proposed methods on helping the model learn to follow instructions.

When L_{pc} is not used, the performance is worse. Especially, compared to MOCA, the task success rate in the unseen settings drop from 3.0 to 1.7. It is aligned with our hypothesis that the alignment between the instruction and the other modalities is hard to learn by using the original training objective alone. On the other hand, though the fine-grained program counter cannot improve the performance



(a) MOCA (b) MOCA + PC (c) MOCA + PC + L_{pc}

Figure 1: Attention map of the MOCA baseline and proposed models. Y-axis is instruction words, and X-axis is the corresponding instruction index of each action. When L_{pc} is used, the attention is more concentrated to the corresponding instruction words. On the other hand, adding program counter alone does not cause much difference compared to the original one.

in the seen setting, when τ is used, it is able to boost the performance in the unseen setting, achieving the best results on the goal success rate. It suggests that the program counter and L_{pc} could be more important in the unseen setting, and the fine-grain program counter could possibly further improve the model's generalization capability.

7.2 Boundary Adherence Score

Here, we analyze the proposed methods with BAS and show results in Table 4. From Table 4, we can see both the attention-based and the gradient-based scores are higher when our L_{pc} is used. This matches our motivation that better alignment can improve the performance. On the other hand, when the fine-grained program counter is used, the adherence score is lower than the coarse-grained ones. One possible explanation is that errors accumulate more severely since there are more steps in the fine-grain setting.

Another interesting observation is that when the program counter is used, the gradient-based scores differ from the attention-based scores more. Especially, when the oracle counter is used, even though the attention-based score is nearly perfect, the gradient-based scores are only around 0.70. It suggests that some information across the instruction boundaries is carried by the LSTM encoder layer, and such information may be essential to the model.

7.3 Qualitative Analysis and Examples

To further understand how the proposed program counter works, we do qualitative analysis by drawing the attention map in Figure 1. We simply use the first sample in the valid-seen split of the data to draw the figures. The Figure 1a shows the attention map of the vanilla MOCA model. We can observe that the MOCA model already has some extent of the ability to align instructions and actions without any modification, which is coherent with the analysis of the proposed BAS on baselines in Section 4. The Figures 1b and 1c are the attention maps of the MOCA + PC model and the MOCA + PC + L_{pc} model respectively. Compared to the vanilla MOCA model, the attention distributions generated by the models with PC and L_{pc} are more concentrated to the corresponding instruction words. On the other hand, there is no very significant difference between the attention maps of MOCA model and MOCA + PC model. Therefore, we can draw a similar conclusion to Section 7.1 that it is hard to learn the alignment between modalities by using only the original training objective.

8 Future Work

It is promising to incorporate our program counter module and L_{pc} in the very recently proposed models. For example, our approach can be used in Blukis et al. (2021) for leveraging the information from step-by-step instructions. They can also be used in Kim et al. (2021) as a replacement of the attended visual features. Zhang and Chai (2021)

executes the step-by-step instructions in order explicitly. It will be interesting to analysis their alignment with our proposed intrinsic metrics. We leave the explorations for future work.

9 Conclusion

In this work, we investigate the model's ability to align different modalities in the vision and language navigation task ALFRED. With the proposed Boundary Adherence Score, we find the existing models Seq2Seq and MOCA fail to align the instruction well with other modalities when predicting the action. Therefore, we further propose the *Neural Program Counter* and the auxiliary loss L_{pc} to help the model learn better multi-modal alignments. With our proposed methods, we can outperform the state-of-the-art released model MOCA.

To sum up, our contributions include 1) We propose the intrinsic metric BAS score, which can serve as an analysis tool for modality alignment. 2) We discover that previous models do not align the modalities well. 3) We propose the program counter model as well as the L_{pc} auxiliary loss, and outperform the previous strong baseline.

Acknowledgments

We would like to thank Yonatan Bisk for his in-depth discussions. We are also thankful to the anonymous reviewers for their comments on the paper.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

Jacob Andreas and Dan Klein. 2015. Alignment-based compositional semantics for instruction following. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1165–1174, Lisbon, Portugal. Association for Computational Linguistics.

I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 2016. 3d semantic parsing of large-scale indoor spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1534–1543.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62.

Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. 2016. Deepmind lab. *arXiv preprint arXiv:1612.03801*.

Valts Blukis, Ross A Knepper, and Yoav Artzi. 2020. Few-shot object grounding and mapping for natural language robot instruction following. *arXiv preprint arXiv:2011.07384*.

Valts Blukis, Chris Paxton, Dieter Fox, Animesh Garg, and Yoav Artzi. 2021. A persistent spatial semantic representation for high-level natural language instruction execution. *arXiv preprint arXiv:2107.05612*.

Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12538–12547.

Rodolfo Corona, Daniel Fried, Coline Devin, Dan Klein, and Trevor Darrell. 2020. Modularity improves out-of-domain instruction following. *arXiv preprint arXiv:2010.12764*.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. Iqa: Visual question answering in interactive environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4089–4098.

S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. 2017. Cognitive mapping and planning for visual navigation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7272–7281.

Karl Moritz Hermann, Mateusz Malinowski, Piotr Mirowski, Andras Banki-Horvath, Keith Anderson, and Raia Hadsell. 2020. Learning to follow directions in street view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11773–11781.

Ronghang Hu, Daniel Fried, Anna Rohrbach, Dan Klein, Trevor Darrell, and Kate Saenko. 2019. Are you looking? grounding to multiple modalities in vision-and-language navigation. In *Proceedings of*

the 57th Annual Meeting of the Association for Computational Linguistics, pages 6551–6557, Florence, Italy. Association for Computational Linguistics.

Siddharth Karamcheti, Dorsa Sadigh, and Percy Liang. 2020. Learning adaptive language interfaces through decomposition. In *Proceedings of the First Workshop on Interactive and Executable Semantic Parsing*, pages 23–33, Online. Association for Computational Linguistics.

Byeonghwi Kim, Suvaansh Bhambri, Kunal Pratap Singh, Roozbeh Mottaghi, and Jonghyun Choi. 2021. Agent with the big picture: Perceiving surroundings for interactive instruction following. In *Embodied AI Workshop CVPR*.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. 2020. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision*, pages 104–120. Springer.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. 2019a. Self-monitoring navigation agent via auxiliary progress estimation.

Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. 2019b. The regretful agent: Heuristic-aided navigation through progress estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6732–6740.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. 2016. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*.

Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. 2018. Mapping instructions to actions in 3d environments with visual goal prediction. *arXiv preprint arXiv:1809.00786*.

Takayuki Okatani and Van-Quang Nguyen. 2020. A hierarchical attention model for action learning from realistic environments and directives. *ECCV EVAL Workshop*.

Kunal Pratap Singh, Suvaansh Bhambri, Byeonghwi Kim, Roozbeh Mottaghi, and Jonghyun Choi. 2020. Moca: A modular object-centric approach for interactive instruction following. *arXiv e-prints*, pages arXiv–2012.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749.

Shane Storks, Qiaozi Gao, Govind Thattai, and Gokhan Tur. 2021. Are we there yet? learning to localize in embodied instruction following. *arXiv preprint arXiv:2101.03431*.

Hao Tan, Licheng Yu, and Mohit Bansal. 2019. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2610–2621, Minneapolis, Minnesota. Association for Computational Linguistics.

Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. 2018. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*.

Yichi Zhang and Joyce Chai. 2021. Hierarchical task learning from language instructions with unified transformers and self-monitoring. *arXiv preprint arXiv:2106.03427*.