ALFRED Learns to Speak

Gabriel Moreira* **Vineeth Reddy Vatti***
{gmoreira, vbv}@andrew.cmu.edu

Abstract

Embodied AI is an active research field that focuses on enabling virtual assistants and robots to interact and learn from environmental stimuli. In recent years, several datasets have been proposed in which models are required to navigate and interact with their surroundings. In this report, we present our approach to the ALFRED challenge. Motivated by the advances in joint speaker-follower models for Vision-and-Language Navigation, we put forward a speaker model for ALFRED. We show empirically that our speaker is capable of generating coherent and free-form natural language instructions and thus it may be employed both in data augmentation strategies as well as in joint pragmatic speaker-follower systems. We propose metrics to evaluate the quality of the generated instructions and provide qualitative interpretations of the results.

1 Introduction and problem definition

The field of embodied AI focuses on enabling virtual assistants and robots to interact and learn from environmental stimuli. The problem can be generally described as follows. In order for a follower F, with an encoded representation of its egocentric vision up to an instant T, $y_{1:T}$, to execute a set of actions $a_{1:T}$, a commander (or speaker) should produce an instruction $i_{1:K}$ such that $P_F(a_{1:T}|i_{1_K}, y_{1:T})$ is maximized. Conversely, the follower should choose the sequence of actions $a_{1:T}$ that maximizes $P_C(i_{1:K}|a_{1:T}, y_{1:T})$.

An agent that can perform instructions after taking in natural language as an input can be useful in a multitude of use-case downstream tasks. An example of this are household assistive bots, which can help in performing regular day-to-day activities and can also be used to assist people with disabilities e.g., the visually impaired, by identifying and locating objects. Going beyond the household domain, these agents can also play a vital role in a search and rescue scenarios by navigating hard to reach or otherwise inaccessible areas.

Turn left and walk over to the stove, then look up at the microwave. Open the microwave and put the apple inside, then close the door and turn on the microwave.

Figure 1: Proposed speaker model for ALFRED, with egocentric images obtained from AI2Thor (Kolve et al., 2017) and the instruction generated by our model below.

In recent years, embodied AI has become the subject of active research, with datasets such as ALFRED (Action Learning From Realistic Environments and Directives) (Shridhar et al., 2020), that bridge the gap between vision and natural language instructions with the goal of having a robot complete household tasks. In order for an embodied agent to perform any task, it must be able to define a plausible set of actions, or an action space, and then choose which of those actions is the best to perform based on stimuli. The first step requires an agent to semantically map their environment using the vision modality to identify objects it could pick and trajectories it could take. The second step requires the use of audio or text modality to understand the command from a human, and potentially hold clarification dialog with the commander. In a real-world situation, the embodied agent must continuously analyze both vision and audio/text modalities to continuously infer the next action.

When considering the ALFRED challenge, different techniques have been proposed to solve the embodied task-completion problem. These range

*Everyone Contributed Equally – Alphabetical order

from end-to-end architectures like the Episodic Transformer (Pashevich et al., 2021), to modular alternatives such as FILM (Min et al., 2021). Nevertheless, joint speaker-follower models and the pragmatic reasoning thereby made possible, in spite of achieving good performance in Vision-and-Language Navigation (VLN) (Fried et al., 2018b), have yet to be applied to ALFRED.

In this paper, we set out to address this gap by proposing a speaker model for the ALFRED dataset. The desiderata for our model are two-fold. The speaker must be capable of generating fluent natural language instructions and these instructions should help the follower succeed in the task. Overall we make the following contributions:

1. A sequence to sequence (Seq2Seq) (Sutskever et al., 2014) speaker model architecture whose code is available on GitHub.

2. A follower-based performance metric to evaluate the quality of the generated instructions.

The rest of the report is organized as follows. We begin by reviewing prior work done in VLN, task completion and other datasets which share some similarities with ALFRED. We then briefly describe the dataset and the data used in our implementations. This is followed by our proposed approach, wherein we first describe a baseline to serve as a reference and then lay out the final speaker model. Subsequently, we present the results from our experiments. Finally, we put forward some concluding remarks and future research directions.

2 Related work and background

The baseline model proposed in ALFRED comprises a CNN-LSTM sequence-to-sequence architecture. The follower is trained using imitation learning on expert trajectories, which ensures that the language directives match the visual inputs. The model reweighs the instruction based on the history, and joins the previously executed action with the current observation features. The joined inputs are then passed as an input to the LSTM block to produce the current hidden state, which is then combined with the previous features to predict the pixelwise interaction mask over the observed image and the next action.

We will now review other works in the VLN and task completion literature. Previous VLN algorithms used pre-trained transformers only on encoding the visual features, and needed to re-train

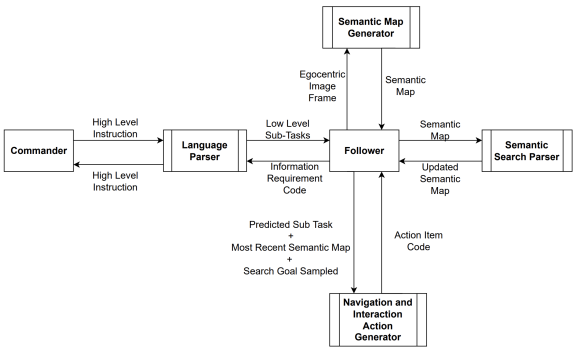


Figure 2: Data flow diagram in FILM

completely for the navigator transformer. Comparatively, VLN-BERT (Hong et al., 2021) proposes training a single transformer for navigation which can also perform as an RNN for time dependent inputs. The overall VLN-BERT model acts as a multi-functional network which encodes the language, performs cross modal grounding, checks the agent's state and makes the decision. It can also multitask to simultaneously address the navigation and remote object grounding. More recent, and generaly used as a benchmark is the Episodic Transformer (Pashevich et al., 2021).

FILM (Min et al., 2021) proposes a modular framework for the embodied task completion, where it is designed to have three major modules - semantic mapping, semantic search, and navigation and interaction. The semantic map generator is responsible for parsing egocentric views of the virtual environment into two dimensional maps that contain information about the location and class of each object in the environment. The semantic search policy module is used to localize objects that are small in size, which are hidden away inside receptacles, or are not identifiable from specific angles. The navigation and interaction module is responsible for either navigating towards an object, interacting with an object, or indicating that a clarification or more information is needed to carry out a task. Fig 2 shows how data flows between all the different modules in FILM.

Learning by imitating, despite its widespread adoption, is not the only approach to solving embodied AI, with the alternative, as shown in (Misra et al., 2017) being reinforcement learning. The authors encode the previous action and instructions using an LSTM, and egocentric images using a CNN. The multimodal concatenation goes through an MLP that is learned by maximizing a reward.

The aforementioned papers focus on modeling the follower agent. However, research has also shown the benefits of joint speaker-follower modeling (Fried et al., 2018a). The authors propose the use of pragmatic reasoning, wherein the model for each agent comprehends a model of the other i.e., the commander has in itself a model of the follower. This allows e.g., the commander to anticipate how the follower would react to a certain utterance by generating several utterances and having its model of the follower selecting the one which produces the desired outcome. The speaker-follower model proposed in (Fried et al., 2018b) builds upon this pragmatic approach in order to navigate the R2R dataset (Anderson et al., 2018). The authors introduce speaker-driven data augmentation and let the follower pragmatically choose the best sequence of actions.

2.1 Related datasets

Given our goal to develop a speaker model for ALFRED, for which there is no prior work, we review in this section datasets wherein joint speaker-follower modeling has been used.

SAIL is one of the earliest decision-based task-completion benchmarks, with just three environments and less than 1k instructions. This was the dataset used for the pragmatic models in (Fried et al., 2018a). In VLN, Room-To-Room (R2R) and RxR are two flagship datasets which use realistic imagery from Matterport3D. The R2R benchmark (Anderson et al., 2018) contains 90 RGB-D environments and over 20k instructions for room navigation. This was the environment of choice for the speaker-follower model proposed in (Fried et al., 2018b), which as we will show, will be the basis for our own speaker model. R2R's successor, Room-Across-Room (RxR) (Ku et al., 2020) goes one step further, collecting +100k multilingual instructions.

Finally, as a successor of ALFRED, TEACh (Padmakumar et al., 2021) is a task-completion benchmark taking place in AI2Thor environments as well. What sets it apart form the former is that it presupposes by default both speaker and follower models in order to accomplish the tasks. The speaker is not only required to give the starting instruction, but there should also be free-form dialogue and collaboration between the agents. We believe our work on a speaker model for ALFRED may also shed light on how to best model a speaker in TEACh.

	Matterport		THOR	
Model	Seen	Un	Seen	Un
Full Model	**27.1**	19.6	**77.7**	**18.08**
Baseline	15.9	16.3	2.18	1.54
Actions	18.5	17.1	4.53	2.88
Actions + Vision	21.2	16.6	35.6	7.50
Actions + Language	23.0	**22.1**	4.03	3.64

Table 1: Navigation success (%). Unimodal vs Multimodal Analysis for VLN in R2R adapted from (Thomason et al., 2019).

2.2 Unimodal approaches

In general, embodied AI tasks are difficult to perform in an unimodal fashion. Nevertheless, as reported by (Thomason et al., 2019), dataset biases in R2R can make unimodal models surpass fully multimodal baselines, as shown in Table 1. Such is the case for unimodal models with access to only actions or actions and vision and no instructions. As explained by the authors, this is due to the ability of unimodal agents to exploit topological biases and correlations in indoor environments. While such biases are also present in ALFRED, we expect the longer episodes and complex tasks to hinder the performance of unimodal baselines.

2.3 Relevant techniques

From prior work conducted on similar datasets, we identified several techniques that may yield positive results if translated to ALFRED.

Pragmatic reasoning The use of pragmatic reasoning, allowed for considerable performance gains in SAIL. We posit that endowing the commander and follower of ALFRED with pragmatic reasoning may yield less ambiguity in instruction generation by the commander and in its interpretation by the follower.

Data augmentation In order to help the follower learn, (Fried et al., 2018b) train the commander first on ground-truth data and then use it to supervise the follower by synthesizing new tasks and their respective instructions.

Object-based features Replacing CNN features by object-based representations to help the model generalize to unseen environments. In (Hu et al., 2020), the authors posit that CNN-based features in VLN may hinder generalization since the model may ground the instruction to a specific object from a specific environment. They propose to replace

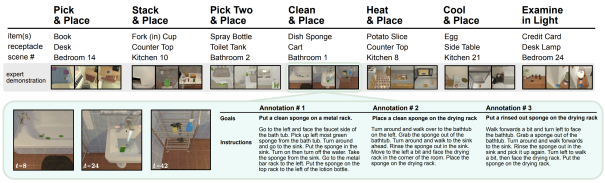


Figure 3: ALFRED annotations

	Val Seen	Val Unseen
Annotations	820	821
Scenes	108	88

Table 2: ALFRED Validation data splits

these features by object label and location-derived features.

3 Task setup and data

ALFRED contains 25k human language annotations, 8k demonstrations, 2k unique tasks, 120 scenes, 56 object classes and 26 receptacle classes. The benchmark also provides high level and low level step-by-step goals to follow a particular trajectory and perform an action. It allowed a much richer interaction with the environment by utilizing the AI2Thor simulator (Kolve et al., 2017).

In Figure 3 (Shridhar et al., 2020) we can see the high level agent tasks that are available in ALFRED. All of the actions are characterized by the object in focus and the destination receptacle (barring Stack and Place where we use a base object). As we can see from the image, ALFRED contains expert demonstrations of 7 different tasks, while using 58 object classes, 26 receptacle classes, and 120 different indoor scenes combinations. For each task parameter, three different expert demonstrations are created by starting the agent and the object in three random locations. Although the places of spawn are not completely random, they are logical i.e., you can find an knife in a drawer but not an apple. Thus from 2685 combination of task parameters, we obtain a total of 8055 unique demonstrations with an average of 50 action steps. For this use case we majorly test on the validation split whose data split can be found in Table 2.

3.1 Task Setup

Given the task dictionaries provided in the ALFRED dataset (available on GitHub) our speaker model will receive as input the sequence of actions and ResNet-18 features available by downloading the `json_feat_2.1.0` folder. The actions we use are the 12 low-level actions used in the AI2Thor API calls.

Low-level actions `Look Down`, `Move Ahead`, `Rotate Left`, `Look Up`, `Slice Object`, `Rotate Right`, `Open Object`, `Put Object`, `Close Object`, `Toggle Object On`, `Toggle Object Off`, `Pick up Object`

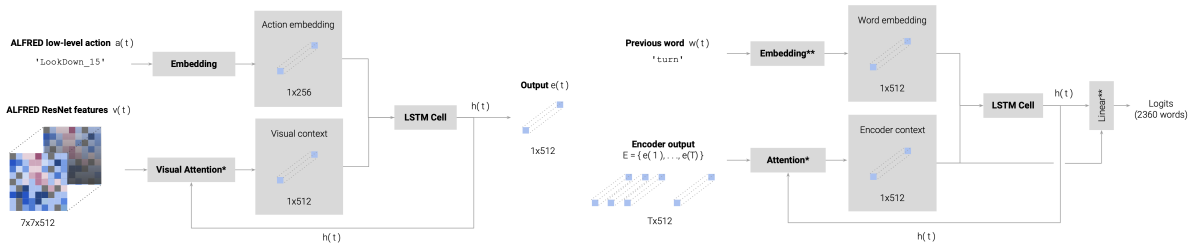
These are stored in the task dictionaries under the keys `plan` → `low_actions`. The corresponding value is a list. The actual low-level action words are under the keys `discrete_action` → `action` for each element of the aforementioned list. These actions are then indexed to ALFRED's action vocabulary which is available in the repository (`pp.vocab`). For the reference instructions, we use the already vocabulary-indexed instructions available in the task dictionary under the keys `num` → `lang_instr`.

Finally, when batching indexed actions sequences with different lengths, we pad them with zeros, which is the action vocabulary index of the `<PAD>` token. We do similarly for the reference instructions. The visual feature tensors, when stacked to form batches are padded with zeros as well.

4 Baseline speaker

Since there was no prior work, in our speaker approach to the ALFRED challenge we begin by describing our baseline model, which is an adaptation of the model proposed in (Fried et al., 2018b). We formulated the speaker as an agent which combines information about the world and the actions taken by a follower, in order to generate a natural language instruction of the task to be completed. To this end, in each episode, the speaker has access to a time sequence of visual frames/features and a sequence of actions taken by the follower. These do not inform about the objects interacted with, however.

In order to create a multimodal speaker baseline, we adopted the LSTM encoder-decoder with attention proposed in (Fried et al., 2018b). In each episode, the encoder is fed the sequence of ALFRED's ResNet-18 features $\mathcal{V} = \{v_t\}_{t=1,...,T}$, where $v_t \in \mathbb{R}^{7\times7\times512}$, plus the sequence of low-level API actions $\mathcal{A} = \{a_t\}_{t=1,...,T}$ for that episode, indexed to ALFRED's action vocabulary. As shown in fig. 4a, the speaker has a visual atten-



(a) Speaker encoder

(b) Speaker decoder

Figure 4: Proposed sequence-to-sequence ALFRED speaker model.

tion module, with the query being the LSTM cell's hidden state $h_e(t-1)$ from the previous time step, and the keys being the 512-d vectors from the 7×7 ResNet features grid. This visual attention block is essential for the speaker to identify and attend to the small objects and the respective parent receptacles that should be interacted with. The output of the encoder obtained by simply stacking of all the LSTM's outputs over time.

Once the encoder has iterated through all the frames in the input, the decoder will then be fed the corresponding output along with the previous decoded word. The speaker's decoder is another LSTM cell with a context soft-dot attention block attending to the encoder's output as shown in fig. 4b. The query, in this case, is the LSTM cell's hidden state from the previous time step $h_d(t-1)$. Unlike the attention block of the encoder, this module is the means by which information is shared from the encoder to the decoder and it allows the decoder to determine which subsection of the action sequence is relevant to infer the next word. These words are predicted by feeding the LSTM's hidden states to a hidden linear layer with a Tanh activation, followed by the output layer with the dimension of ALFRED's word vocabulary (2360 words) and a log-softmax activation.

Since the model from (Fried et al., 2018b) was trained using the R2R dataset, we adapted the architecture and trained the speaker from the bottom-up in ALFRED. The speaker was trained with a teacher-forcing probability of 1.0 on ALFRED's training split, which comprises 21,023 episodes, by minimizing the negative log-likelihood. We employed stochastic gradient descent for 126k iterations with learning rate set to 0.001 on a p3.2xlarge AWS instance. Dropout was set to 0.1. To keep the baseline simple, at inference time, we used greedy search to perform the decoding.

Unimodal baselines As a unimodal baseline for the speaker, we used the aforementioned Encoder-Decoder model but ablated the visual input. While there may be natural occurring correlations in the dataset that allow the speaker to generate instructions when given only a sequence of actions, we expect many of these generated instructions to be either wrong or to incorrectly refer to certain objects or elements of the scene.

5 Proposed model

Since there was no prior work on speaker models for ALFRED and given that our baseline was already an implementation of a state-of-the-art speaker model which produced good results, for the final speaker we opted to make minor changes in the baseline and adjust the training configuration. These changes were motivated by the three types of failure modes identified in the baseline:

Poor generalization The speaker had a considerably worse performance in unseen environments (difficulty generalizing), often misidentifying not only small but also large objects with appearances dissimilar to those the model had learned e.g., misconstruing a white safe for a white desk, or a tub for a counter. We posit that these were likely due to overfitting the training set since e.g., there were many examples of storing credit cards in safes, thus the model would tend to predict safe more often when the target object was a credit card.

Text degeneration In some instructions, the synthesized text would degenerate into repeated patterns. We justify this with the fact that the baseline had minimal training.

Minimalistic instructions Some instructions are fluent and semantically correct but do not provide the necessary detailed steps of how to accomplish a task. This is arguably the most insidious of the

	Baseline	Final
Optimizer	SGD	AdamW
Starting LR	10^{-3}	10^{-3}
Scheduler	-	Exponential (0.98)
Iterations	10^5	10^6
Teacher-forcing	1.0	$0.9 \rightarrow 0.75$
Decoder	Greedy	Beam-width 20

Table 3: Comparison of the training setup between the baseline and the final model.

failures modes since it is difficult to avoid. Part of the reason as to why this happens may be attributed to the suboptimal greedy decoding. Nevertheless, we also identified that many reference instructions in ALFRED were not appropriate e.g., 'Pick up the apple' instead of 'Turn towards the counter on your left. Pick up the apple on the left side of the plate'.

With the goal of improving generalization, we added shared weights in the decoder between the embedding layer of the previous word and output linear layer. Dropout was also increased from 0.1 in the baseline to 0.3 in the final speaker. Additionally, we introduced a teacher-forcing linear scheduler starting with a teacher-forcing probability of 0.9 and decreasing until reaching 0.75, at which point it stayed constant (recall that in our baseline we had used pure teacher-forcing). Having a teacher-forcing probability lower than 1.0 ensures that the model will eventually be fed its own predictions during training. This in turn, promotes stability and prevents sentence degeneration. Finally, in order to generate longer and more detailed sentences, we replaced the greedy search used in the baseline by beam-search.

Akin to the baseline, for generating the sentences that are similar to the ground truth we kept the cross entropy loss, as it makes sure that the structural similarity is maintained. The detailed training configuration is shown in Table 3.

6 Experimental evaluation

In this section, we set forth the experimental setup used to assess the performance of our speaker model, as well as the quantitative results obtained. Attending to the lack of literature on the matter, we dedicate some paragraphs to the problem of quantifying the quality of an instruction, present our choices of performance metrics and justify them.

6.1 Setup

In order to evaluate the quality of the instructions generated by our speaker model, let us begin by characterizing what is, in fact, a good instruction. This is not at all an objective definition since e.g., an instruction may be fluent, semantically correct and yet incapable of leading the follower towards the completion of the desired task. In the context of ALFRED, a good instruction should be a fluent and free-form natural language description of the task, grounding its atomic elements to the visual data, so as to aid the follower. As such, the instruction '*go straight towards the counter, look up to the microwave, put the apple inside the microwave and heat it*' is assumed to be preferable than the alternative '*heat the apple in the microwave*'. Both are fluent descriptions of the task, but we posit that the later would present greater difficulty for the follower.

The problem with the aforementioned definition of a good instruction is that there is no predefined metric to quantify the traits we just identified as being desirable. Therefore, we must use proxies e.g., the corpus-level BLEU score and the task success rate. While the former is straightforward to compute, by itself it is an ill-conceived metric for evaluating instructions since it is not indicative of how easy they are to follow. This was one of the findings of (Fried et al., 2018a). To gauge instruction-*followability*, we need a follower and its task success rate when fed the synthetic instructions. To this end, we rely on the modularity of FILM's follower. This model only uses the instruction in the language processing module to identify task types, objects and parent objects. Thus, we posit that the correct identification of all three attributes is a necessary condition for success. While FILM's language module is not perfect, we can establish a performance ceiling since the ground-truth attributes are given in ALFRED's task JSON files. Thus, to ascertain instruction followability we feed both the synthetic and reference instruction to FILM, obtain the three aforementioned attributes and compare them against the ground-truth.

6.2 Results

We conducted experiments in order to quantify the ability of FILM's follower to identify the task type, the objects and its parents from our generated instructions. We present the accuracies obtained for the baseline (base) and for the final speaker model

	Accuracy (%) ↑ (val seen)			
	Base	Final (G)	Final (BS)	Ref.
Task Type	82.8	90.7	90.2	99.0
Objects	78.5	89.0	85.8	95.1
Parents	68.3	87.8	87.5	96.2

Table 4: Accuracy evaluated using FILM's LM (`valid_seen` split).

	Accuracy (%) ↑ (val unseen)			
	Base	Final (G)	Final (BS)	Ref.
Task Type	75.7	91.7	88.8	99.4
Objects	68.6	77.2	77.5	93.8
Parents	57.0	55.1	56.5	96.8

Table 5: Accuracy evaluated using FILM's LM (`valid_unseen` split).

both with greedy (G) and beam-search (BS), in Tables 5 (validation seen) and 4 (validation unseen). In addition, since FILM's language module is not a perfect classifier, we also present the accuracies for the reference instruction (Ref). Despite the unreliability of BLEU scores, for the sake of completeness we show an histogram of the corpus-level BLEU scores in Fig. 5, comparing the baseline with the final model.

We see consistent improvement from the baseline to the final model, in the ability of the follower to correctly infer the task type and the objects. This can be attributed to the changes in architecture, training and decoding mentioned in Section 5. As expected, this improvement is overall much more salient in the seen split. In fact, there was either no change or even degradation in the inference of parent objects in the unseen validation split. We also notice that using beam search instead of greedy results in lower accuracies. As we shall show in the following section, beam search leads to generally longer instructions and with more detail. This may hinder FILM's ability to correctly parse the instruction. Finally, we see an improvement in the average corpus-level BLEU score from 0.17 to 0.22. We may attribute this to the increased accuracy of the final speaker in identifying target objects, as evidenced from Tables 5 (validation seen) and 4.

7 Analysis

In this section we expound some of the results put forward in Section 6. We analyze the impact of beam-search, compare instruction lengths, visualize the attention the decoder pays to the encoder's output and conclude with examples of generated instructions and the respective reference.

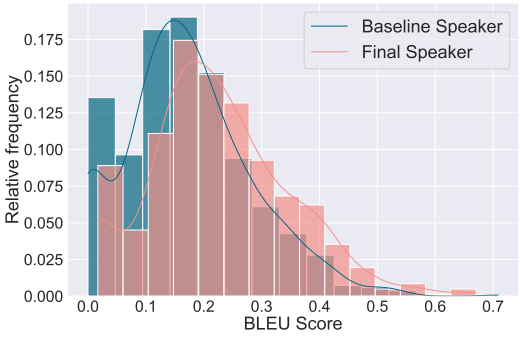


Figure 5: Corpus-level BLEU scores comparison between baseline and final speaker. The baseline has a mean score of 0.17 and the final speaker a mean of 0.22

.

7.1 Visualizing context attention

In Fig. 6 we show an example of the soft-dot context attention distribution. This attention module is used by the decoder to focus on a specific part of the encoded input, for each decoded word. Recall that this encoded input is the output of the encoder's LSTM cell. The same attention was used in (Fried et al., 2018b). We notice that the typical diagonal attention structure often found in seq2seq models is absent in our speaker model. Instead, we observe certain attention clusters centered around actions such as pick-up and put-down. Conversely, the decoder does not attend to the elements corresponding to actions such as move-ahead.

7.2 Impact of beam-search

As mentioned in Section 5, we attempted to solve the problem of coarse low-detail instructions by replacing the greedy decoding of the baseline by beam-search. We show in Fig. 7 an histogram of the length difference between the generated instruction and the reference, for the greedy and beam-search alternatives. While the distributions have an almost complete overlap, instructions decoded using beam-search have, on average less 15 tokens than the reference while greedy-decoded instructions have, on average, less 57 tokens. An example comparing a greedy instruction against a beam one is shown in Table 6. The generated instructions are almost identical with the exception of two cues

Figure 6: Visualization of the soft-dot attention from the decoder to the outputs of the decoder. Actions on the vertical axis are the inputs to the encoder. Decoded instruction is shown in the horizontal axis (zoom-in): '*Turn right and walk to the tv. Pick up the remote control from the tv stand. Turn around and walk to the box on the table. Put the remote control in the box on the table. Pick up the box from the table. Turn around and walk to the coffee table. Put the box down on the coffee table.*'.

Reference	Turn around and face the wall with the bat on the floor pick the baseball bat up off of the floor turn around and maneuver of to the desk at the opposite corner of the room turn the lamp on on the desk
Greedy	Turn right and walk to the baseball bat on the floor. Pick up the baseball bat from the floor. Turn right and walk to the lamp on the desk. Turn on the lamp.
Beam search	Turn right and walk to the baseball bat on the floor. Pick up the baseball bat from the floor. Turn right and walk to **the other side of the room**. Turn on the lamp on the desk.

Table 6: In general, beam-search produces more detailed instructions.

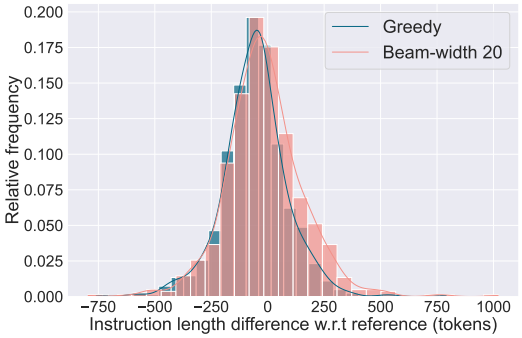


Figure 7: Greedy decoded instructions have on average, less 57 tokens than the reference instructions whereas using a beam-width of 20, this mean difference is 15 tokens. Data from `val_unseen` split.

which are likely vital for the follower. The beam-search instruction mentions '*walk to the other side of the room. Turn on the lamp on the desk*' while the greedy one simply says '*walk to the lamp on the desk*', without specifying where the lamp is. As we had shown in Section 6.2 however, beam-search did not contribute to increase FILM's accuracies. We argue that there may be a trade-off between instruction length and complexity and how easy it is to process. One instruction may be short and contain all the objects necessary for the language module to have an accuracy of 100%, yet it may not be human-like.

7.3 Qualitative analysis and examples

We conclude our qualitative analysis by presenting concrete examples of instructions generated from the baseline and follower speakers (Tables 7, 8). We can see from the tables how the final speaker is able to provide more clues compared to the baseline, like 'close the door' as shown in 8. In Table 9, we can see all the generated instructions that have been correctly classified (tasks, objects and parents) by the FILM's language module. In contrast, in Table 10 we show examples where at least one classification has been unsuccessful by FILM. This can arise either because the generated instruction might completely miss the object to focus on, or if the generated instruction has a slight different initial action type. We can see e.g., there are cases where the generated instruction asks the agent to '*turn right*' while the reference instruction is to '*turn around*'. See Appendix for more examples.

To complete our analysis, we also wanted to test our performance metric which consisted of employing FILM's language module to evaluate our speaker model. In Table 11 we provide examples, where FILM itself is unable to classify the ground-truth. From the example we can see that it is usually in the case where the objects are very similar e.g., '*saltshaker*' and '*peppershaker*'.

Reference	Turn right and walk over to the counter left of the sink open the bottom cabinet and take the red rag from it turn around and walk to the bathtub put the red rag into the bathtub.
Baseline	Turn right and walk to the white cabinet. Open the cabinet door, pick up the red cloth from inside and close the door. Turn right and walk to the white table. Put the red cloth on the middle of the white table.
Final	Turn right and walk to the sink. Open the cabinet under the sink and take out the red rag. Turn around and walk to the tub. Place the rag in the tub.

Table 7: Comparison between baseline and final speakers shows how some incorrectly identified parent objects in the baseline are correctly identified by the final speaker.

Reference	Turn left and go forward to the stove. Pick up the pot on the right side of the stove. Turn around and go straight, then turn left to go to the fridge. Open the fridge, put the pot inside and chill it, then take the pot out. Turn around and go to the white table. Put the pot on the white table.
Baseline	Turn left and walk to the stove. Pick up the pan on the stove. Turn around and walk to the fridge. Chill the pan in the fridge. Turn left and walk to the table. Place the pan on the table.
Final	Turn left and walk to the stove. Pick up the pan on the back right burner. Turn around and walk to the fridge. Place the pan in the fridge, close the door, wait a moment and take it out again. Turn around and walk to the white table. Place the pan on the table.

Table 8: Comparison between baseline and final speakers shows how the final speaker gives more clues e.g., '*close the door*' and specified the color of the table.

8 Future work and limitations

The proposed speaker model is capable of generating free-form instructions which generally, can be deemed qualitatively correct. In fact, the changes put in place from the baseline model to the final speaker led to a substantial increase in the ability of FILM's language model to process the generated instructions, as shown in Section 6.2. There are however, certain lagging areas that should be highlighted. For instance, instruction evaluation can still be improved. FILM's language model seemingly solved this problem. Nevertheless, instructions containing just the correct objects, parents and actions and lacking other information may still be correctly processed by FILM, despite them not being correct instructions as per the definition laid out in Section 6.1. An ideal evaluation metric, would be to setup a follower who can take all the instructions as an input and can try to complete the tasks accordingly. In thus scenario, the Success Rate (SR), or Goal Completion (GC) can be some of the metrics that would give an actual understanding of how good the speaker model is performing without the variability of the Language Parser as in our case. In addition to this persisting evaluation problem, in unseen environments the speaker still produces semantic errors in what can be considered large objects e.g., a microwave. Given FILM's success in addressing the object detection problem, a possible strategy for tackling it could involve the use of Fast-R-CNNs and semantic maps instead of ALFRED's ResNet-18 features. With respect to the field of Embodied AI itself, there is a lack of research in the field of a multi agent set ups which could mimic the real world scenario, and can be useful in solving a plethora of new tasks. This limitation can be majorly attributed to the lack of simulators with multi-agent features until recently.

9 Ethics

This section discusses the major risks and ethical challenges involved in developing and deploying embodied task completion agents.

Product Risks Incorrect predictions and actions can be severely harmful and even life-threatening in some scenarios. For instance, if an agent assisting a person with vision impairments provides incorrect actions, it can lead to accidents. Similarly, an agent navigating incorrectly can collide with humans, or other objects leading to loss of public order, safety risks and potential economic losses.

Regulatory Risks Regulatory restrictions could be a serious concern for embodied AI tasks in the future. If there are increased cases of inconvenience to the public due to agents operating especially in outdoor environments, or instances where human safety is compromised and economic losses are incurred, it might not be surprising to see new laws restricting the overall use of these agents. As we progress towards building an embodied AI system that can be deployed in the real world, we plan to add checks and balances to our algorithm so that risks to human life and damage to surroundings is minimized. Another potential issue could be data security wherein the data gathered by the agents such as semantic maps could be misused if made publicly available. The community is working on creating robust systems to address these privacy concerns, and we will be incorporating them in our solution.

Business Risks The risks mentioned above falling under Product and Regulatory risks eventually culminate into business risks for embodied agents. Any issues regarding the correctness and efficiency of agents could adversely impact widespread adoption. In addition, issues with the law and regulations would discourage people from buying these agents, hampering business.

Generated instruction ✓	Reference ✓
Turn around and walk to the counter to the left of the stove. Pick up the knife from the counter. Turn left and walk to the fridge. Cut the potato in the fridge into slices. Turn around and walk to the microwave above the stove. Place the knife in the microwave. Turn around and walk to the fridge. Pick up a slice of potato from the fridge. Turn right and walk to the microwave above the stove. Heat the potato slice in the microwave and then remove it. Turn around and walk to the sink. Place the potato slice in the sink.	Turn to the right twice and to to the end of the counter top and turn to the left and go to the end of the counter top take the knife from the counter top turn to the left and go to the front of the refrigerator and turn to the left and go to the refrigerator open the refrigerator door and slice up the potato on the shelf with the knife and close the refrigerator door turn to the right twice and take a few steps and turn to the left and go to the microwave open the microwave door and put the knife in it and close the microwave door turn to the left twice and take a few step and turn to the right and go to the refrigerator open the refrigerator door and take a potato slice from the shelf and close the refrigerator door turn to the right twice and take a few steps and turn to the left and go to the microwave put the potato slice in the microwave and cook it and when finish cooking take out the potato slice from the microwave turn to the right twice and take a few steps and turn to the left and go to the sink put the potato slice in the sink.
Turn right and walk to the toilet. Pick up the pink bar of soap on the toilet tank. Turn around and walk to the sink. Rinse the soap in the sink. Turn around and walk to the tub. Place the soap in the tub.	Walk up to the toilet, then turn right and look down at the small red bin. Pick up the bar of soap out of the small red bin. Turn right and walk forward, then turn left and walk up to the counter. Put the bar of soap in the left sink basin and turn on the water, after a couple seconds turn the water off and remove the now clean bar of soap. Walk over to the tub on your left. Put the clean bar of soap into the tub to the left of the rag.
Turn around and walk to the counter to the left of the stove. Pick up the head of lettuce from the counter. Turn left and walk to the sink. Put the lettuce in the sink, rinse it off, and take it back out of the sink. Turn around and walk to the refrigerator. Put the lettuce in the refrigerator.	Go to the counter left of the fridge pick up the lettuce on the counter go to the sink wash the lettuce in the sink go to the fridge put the lettuce in the fridge.
Turn right and walk to the wooden table. Pick up the large ladle with a black handle. Turn right and walk to the sink on the left. Put the spoon inside the sink, rinse it for a few seconds and pick it back up. Turn around and walk to the wooden table on the right. Put the rinsed spoon on the wooden table in front of the loaf of bread.	Walk straight to wooden table, stop in front of wooden table pick up ladle with black handle from wooden table turn right, walk to kitchen sink, stop at sink put ladle into kitchen sink basin, wash with water, pick up ladle turn around and walk to wooden table, stop in front of wooden table put ladle on table to the right of the loaf of bread.

Table 9: Examples of correctly generated instructions.

Generated instruction ✗	Reference ✓	Failure
Turn around and walk to the counter to the left of the stove. Pick up the knife from the counter. Turn left and walk to the fridge. Cut the potato in the fridge into slices. Turn around and walk to the microwave above the stove. Place the knife in the microwave. Turn around and walk to the fridge. Pick up a slice of potato from the fridge. Turn right and walk to the microwave above the stove. Heat the potato slice in the microwave and then remove it. Turn around and walk to the sink. Place the potato slice in the sink.	Turn right, turn right, walk straight, turn left, walk to the counter pick up the knife on top of the counter turn left, walk straight, turn left, walk straight to the fridge open the fridge, slice the egg inside, close the fridge turn right, turn right, walk straight to the oven open the microwave, put the knife inside, close the microwave turn left, turn left, walk straight, turn right, walk straight to the fridge open the fridge, take out the sliced egg inside, close the fridge turn right, turn right, walk straight to the oven open the microwave, put the sliced egg inside, close the microwave, turn on the microwave. open the microwave, take out the sliced egg inside, close the microwave. Turn right, turn right, turn left, walk straight to the sink put the sliced egg in the sink.	Objects, Parents
Turn left and walk to the sink. Pick up the knife from the sink. Cut the tomato in the sink into slices. Turn right and look up at the microwave. Put the knife in the microwave. Turn around and walk back to the sink. Pick up a slice of tomato from the sink. Turn right and walk to the fridge. Chill the slice of tomato in the fridge and then pick it back up. Turn right and walk to the sink. Place the slice of tomato in the microwave.	Turn left then face the sink counter pick up the bread knife on the left counter slice the tomato on the sink turn right then face on your left to the microwave open the microwave put in the knife in the microwave, close the microwave look down and step on the side to face the tomato pick up a slice tomato turn right and head to the fridge open the fridge then put in and out the fridge then close it turn right then then back to the microwave open the microwave and put in the slice tomato then close it.	Tasks
Turn right and walk to the toilet. Pick up the yellow spray bottle from the toilet tank. Turn right and walk to the toilet. Place the spray bottle on the toilet tank. Turn around and walk to the toilet. Pick up the spray bottle on the toilet tank. Turn around and walk back to the toilet. Place the spray bottle on the toilet tank to the left of the other spray bottle.	Turn around in the bathroom walk over to the counter grab the yellow spray bottle and walk it over to the toilet put the yellow spray bottle on the back of the toilet turn around and go back to the counter look at the other spray bottle pick that spray bottle up and carry it to the toilet place that spray bottle on the back of the toilet with the other one.	Tasks
Turn left and walk to the wall then turn left and walk to the wall then turn left and walk to the wall then turn right and walk to the wall then turn left. Pick up the pen that's in front of you. Turn left and walk to the wall then turn right and walk to the night stand. Put the pen on the plate that's in front of you. Pick up the plate with the pen on it. Turn left and walk to the wall then turn right and walk to the night stand. Put the plate on the night stand.	Turn around and then to the right across the room. Turn right again towards the large windows, and look to the desk. Pick up the watch from the right side of the desk. Place the watch in a bowl on the left side of the desk. Pick up the bowl and watch from the desk. Carry the bowl and turn left to go around the furniture and to the other room. Turn right in between the beds to find the small table. Place the bowl and watch on the table.	Objects

Table 10: Examples of generated instructions that FILM failed to interpret.

Reference	Ground Truth Object ✓	Predicted Object ✗
Grab the vase from the cabinet. Turn on the lamp. Turn left walk to the cabinet. Open the second door from the right take the vase out. Turn right walk to the corner. Turn on the lamp on the end table.	Vase	Glassbottle
Place the salt shaker from the wooden shelf into the cabinet. Turn around and head towards the wooden shelves. Pick up the salt shaker on the top shelf. Turn around and head towards the fridge to your left. Open the lower cabinet to the left of the fridge place the shaker in it and close the door	Saltshaker	Pepperphaker
Put a clean plate on the counter. Turn right and go towards the dishwasher then turn left to face the counter next to the dishwasher. Pick up the plate on the counter. Turn around to go to the sink. Clean the plate in the sink. Turn around to face the counter above the dishwasher. Put the plate on the counter	Plate	Pan

Table 11: Examples of reference instructions that FILM failed to interpret.

Team member contributions

Gabriel Moreira contributed with baseline and proposed model descriptions, the speaker implementation. Shared effort on the introduction, results and analysis.

Vineeth Reddy Vatti contributed with evaluation of the speaker model using the FILM's Language Processor. Added the ethics module. Shared effort on the introduction, results and analysis.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sunderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. 2018. Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

Daniel Fried, Jacob Andreas, and Dan Klein. 2018a. Unified pragmatic models for generating and following instructions. *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:1951–1963.

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

Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. 2021. VLN BERT: A recurrent vision-and-language bert for navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1643–1653.

Ronghang Hu, Daniel Fried, Anna Rohrbach, Dan Klein, Trevor Darrell, and Kate Saenko. 2020. Are you looking? Grounding to multiple modalities in vision-and-language navigation. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 6551–6557.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. 2017. AI2-THOR: An interactive 3d environment for visual AI. *arXiv preprint arXiv:1712.05474*.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. 2020. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. *EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 4392–4412.

So Yeon Min, Devendra Singh Chaplot, Pradeep Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov. 2021. FILM: Following Instructions in Language with Modular Methods. *arXiv preprint arXiv:2110.07342*.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 1004–1015.

Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gokhan Tur, and Dilek Hakkani-Tur. 2021. TEACh: Task-driven Embodied Agents that Chat.

Alexander Pashevich, Cordelia Schmid, and Chen Sun. 2021. Episodic Transformer for Vision-and-Language Navigation.

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. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 10737–10746.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Jesse Thomason, Daniel Gordon, and Yonatan Bisk. 2019. Shifting the baseline: Single modality performance on visual navigation QA. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:1977–1983.

Figure 8: Example of a correctly generated long instruction for a `valid_seen` task. **Speaker-generated instruction**: 'Turn right and walk to the white cabinet. Open the cabinet door, pick up the green spray bottle, close the cabinet door. Turn around and walk to the toilet. Put the green spray bottle on the toilet tank. Turn left and walk to the trash can. Pick up the green spray bottle in the trash can. Turn around and walk to the toilet. Put the spray bottle on the toilet tank to the left of the other green spray bottle.' **Reference instruction**: 'Turn around and walk over to the right bathroom sink open the left cabinet below the right sink and pick up the green spray bottle turn around and walk to the opposite side of the toilet and turn back to face it put the spray bottle on top of the back of the toilet walk a few feet to the left then turn right and walk forward to the trash bin pick up the green spray bottle from the trash bin turn around and walk back over to the left side of the toilet put the spray bottle on top of the back of the toilet.'. Images from AI2THOR.

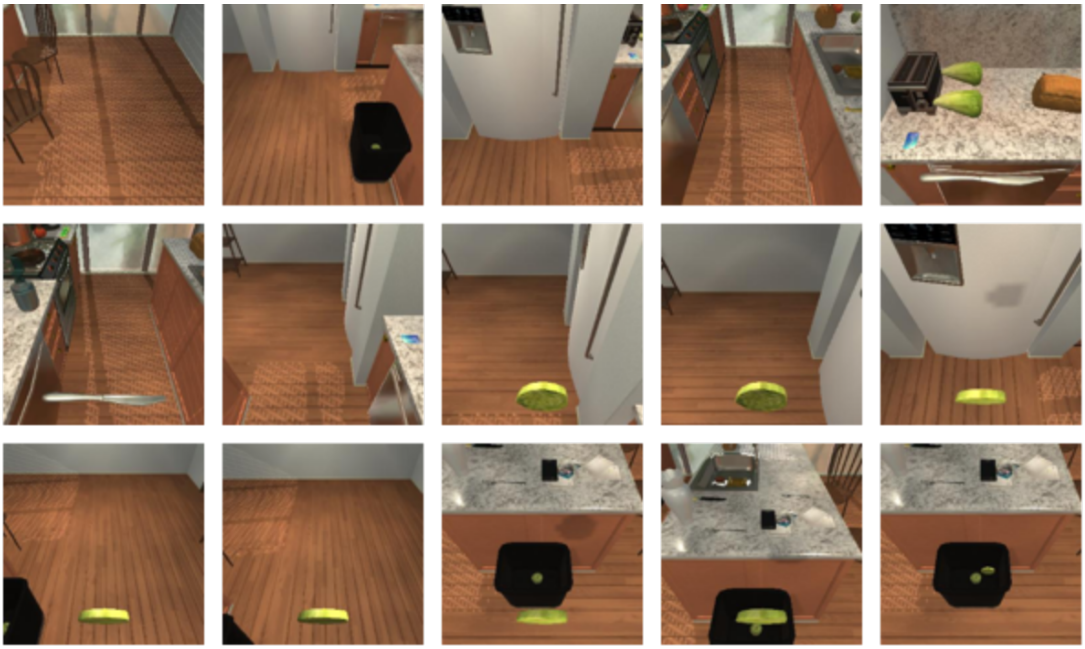


Figure 9: Example of a generated long instruction for a `valid_unseen` task. Note the incorrect labelling of the black bin as a black table. **Speaker-generated instruction**: 'Turn left and walk to the counter. Pick up the knife. Turn left and walk to the counter. Cut the lettuce into slices. Turn left and walk to the other side of the room and face the table. Put the knife on the table. Turn left and walk back to the counter. Pick up a slice of lettuce. Turn left and walk to the fridge. Chill the slice of lettuce in the fridge and then pick the piece of lettuce up. Turn around and walk to the black table. Put the slice of lettuce in the black table.' **Reference instruction**: 'Turn left then right and face the counter with the lettuce on your left. Take the knife from the counter in front of you. Turn left and face the counter on your right. Slice the lettuce in front of you. Turn right then face the counter to your left. Place the knife on the counter in front of you. Turn left then face the counter on your right. Take a slice of lettuce from the counter in front of you. Turn left and face the fridge to your right. Place the lettuce in the fridge for a bit then take it out. Turn around and face the black bin on your left. Place lettuce in black bin in front of you.'. Images obtained using AI2THOR.

Figure 10: Example of a correctly generated long instruction for a `valid_unseen` task. **Speaker-generated instruction**: 'Walk forward, then hang a right and walk over to the kitchen sink. Pick up the green apple out of the sink basin. Turn left and walk over to the stove, then look up at the microwave. Open the microwave and put the apple inside, then close the door and turn on the microwave, after a couple seconds open the microwave and remove the apple, then close the door. Turn left and walk over to the fridge. Open the fridge door and put the heated apple inside, then close the door.' **Reference instruction**: 'Go straight, then turn right towards the sink. Stop in front of the sink. Pick up the green apple from the sink. Turn around, go in front of the stop top. Put the apple in the microwave, turn on the microwave, remove the apple from the microwave. Turn left, go in front of the refrigerator. Put the apple in the refrigerator in front of and to the left of the bread.'. Images obtained using AI2THOR.