CP3106 Project Report

Vision-Language Navigation with Q&A Interactions

By

Han Yuxuan

Department of Computer Science

School of Computing

National University of Singapore

AY 2019/2020

Semester II

CP3106 Project Report

Vision-Language Navigation with Q&A Interactions

By

Han Yuxuan

Department of Computer Science

School of Computing

National University of Singapore

AY 2019/2020

Semester II

Advisor: Assoc Prof. Terence Sim

Deliverables: Report 1 Volume

Abstract

Based on an existing study of VNLA (vision-based navigation with language-based assistance), we present a new learning and helping strategy featuring Q&A interactions. The scenario is as follows, (1) a requester gives an agent a command of finding an object in natural language form; (2) the agent receives the command, explores in a photorealistic environment by its egocentric vision, and tries to navigate to the object; (3) the agent may ask questions during this process, and an advisor will provide help by answering the question; (4) the answers, also in natural language form, will be given to the agent to help its navigation. We define a set of questions, train the agent by imitation learning, develop specific metrics, and compare our solutions with the original work. Results show that even though our solution gives a worse performance than VNLA, as our vague and straightforward answers are much less informative, the agent still has learned to ask reasonable questions and leverage the feedback to improve its navigational decisions.

Code available at: https://github.com/xuange666/NUS-CP3106

Video demo available at: https://youtu.be/E7eLIXqEhCQ

Subject Descriptors:

•Computing methodologies ~ Artificial intelligence ~ Computer vision ~ Computer vision tasks ~ Activity recognition and understanding

•Computing methodologies ~ Artificial intelligence ~ Natural language processing ~ Information extraction

Keywords:

Embodied AI, active learning, navigation, computer vision, natural language processing

Table of Contents

Abstract 3

Table of Contents 4

I. Introduction 5

II. Related Work 8
- 2.1 Embodied Cognition 8
- 2.2 Internet AI & embodied AI 8
- 2.3 Active Learning 8
- 2.4 Multi-modalities 9
- 2.5 Common practices in embodied AI 10

III. Method 11
- 3.1 Characters 11
- 3.2 Models 15
- 3.3 Training and Inference 16

IV. Experiment 18
- 4.1 Setup 18
- 4.2 Results 20

V. Conclusion and Future Work 24

VI. References 25

I. Introduction

One of the long-standing challenges for AI is to follow human's instructions to solve real-world problems. An ideal home robot, for example, should be able to handle a user's requests like "Hey, can you bring me my laptop on the desk of my bedroom?" Clearly, this is an effortless task for us humans, even if we have never been to that home, as we have common sense about the layout of a bedroom, the appearances of desks and laptops, the indoor structure of a house, etc. For AI, however, it still remains a challenging embodied learning problem with multi-modalities: the AI must learn to understand language, interpret visual content, integrate all sources of information, and finally map them to actions.

A large number of related tasks and solutions have been proposed in recent years [4][5][6][9][12][17], with different constraints and problem settings. Among which, VNLA [17] proposes to help the AI agent in the goal-driven navigation by providing language support. Specifically, in VNLA, an agent is spawned at a random corner of a house and given an instruction in the format *"Find <object O> in (one of the) <room R>"*. During exploration and navigation, the agent can signal the need of help, which is handled by an advisor. The advisor has complete information about the environment, who examines the agent's current position to the goal's position, and provides the agent with the optimal actions for the next k steps. See Fig 1 (adopted from [17]) for illustration.

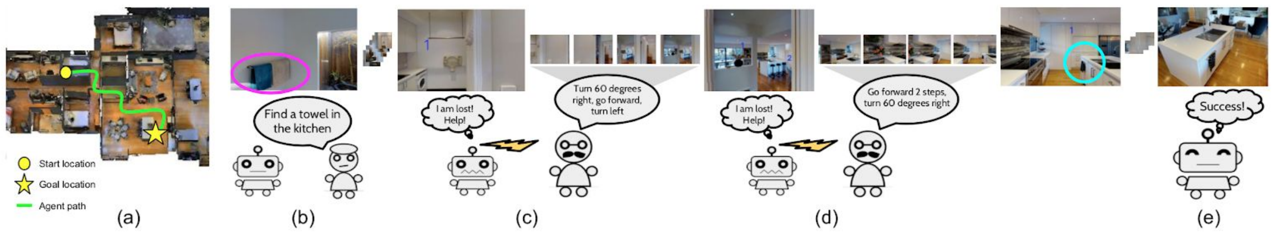
Start location
Goal location
Agent path
(a)
Find a towel in the kitchen
(b)
I am lost! Help!
Turn 60 degrees right, go forward, turn left
(c)
I am lost! Help!
Go forward 2 steps, turn 60 degrees right
(d)
Success!
(e)

Fig 1: VNLA: Vision-based Navigation with Language-based assistance

However, there is still much to be done in VNLA. One obvious shortage is that the interaction between the agent and the advisor is highly restricted and unnatural. The agent, on the one hand, can only say “I am lost! Help!” instead of asking meaningful questions. The advisor, on the other hand, basically tells the agent how to get to the goal with fine-grained action steps, like “turn 60 degrees right, go forward 2 steps, turn left”. We consider this kind of “help” as too direct and over-detailed.

In our work, we change this kind of interaction into Q&A dialogues. The agent can choose questions to ask from our pre-defined question set, and the advisor only gives simple and short answers accordingly. See Fig 2 for illustration. The learning in our settings is more difficult than VNLA, as it is much harder for the agent to utilize the short and straightforward answers compared to the direct descriptions of actions. Besides, the agent not only needs to decide “**whether to ask**”, given the limited “asking budget”, but also decides “**what to ask**”: which questions are the more reasonable questions that could provide more help for navigation under the current situation.

As we train the agent by imitation learning, it is also essential to define teachers that make optimal decisions. The policy of the navigation teacher is straightforward: we calculate the shortest path from the agent current position to the closest goal, and derive actions that move along that path. For the asking teacher, we manually design a set of criterions and mappings to decide “**whether to ask**” and “**what to ask**”. However, these criterions and mappings may not be the “optimum”, as will be discussed in 4.1, but they provide good enough examples for the agent to follow. See details of the teachers in 3.1.

For the following sections of the report, we will first do a review over embodied AI, next introduce our Q&A model and learning strategy, then explain the experiment details and finally summarize the results.

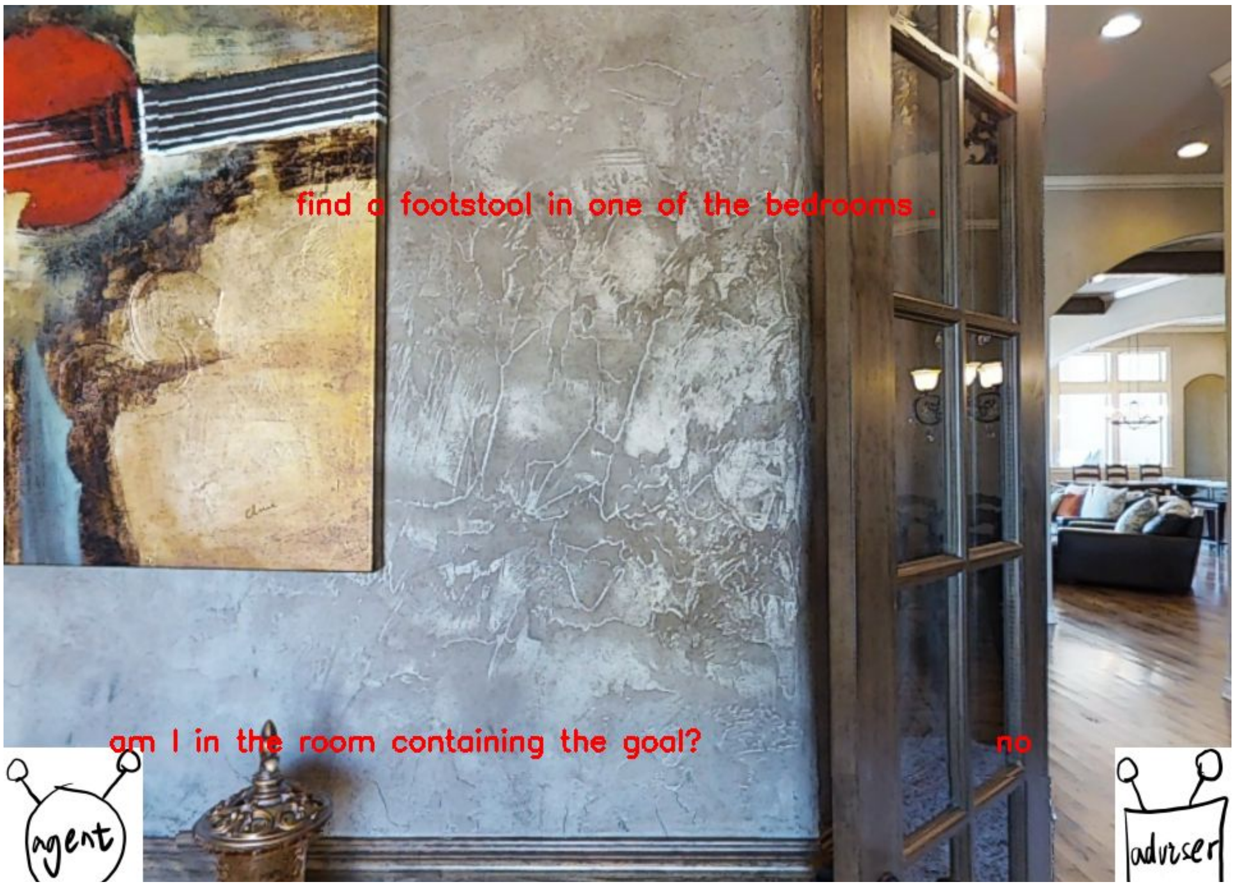
find a footstool in one of the bedrooms .
am I in the room containing the goal?
no
agent
adviser

Fig 2: Q&A interaction in vision-language navigation

II. Related Work

2.1 Embodied Cognition

As is mentioned in a neuroscience blog [15], "Embodied Cognition" is the idea that "the mind is not only connected to the body but that the body influences the mind". In some cases, our mind controls our body on how to behave intelligently, but sometimes the feedback we receive as we interact with the environment (by our body) may also change the way we think (by our mind). An example would be: When frying an egg, our mind tells our body how to break the shell, how often should we turn the egg over, etc. However, after having a taste, our tongue feels it is too salty, which makes our mind learn that we should probably add less salt.

2.2 Internet AI & embodied AI

In the past years, the focus of AI research is mainly on "Internet AI": passive learning on static datasets and making predictions based on acquired knowledge. It can be viewed as a pure "mind controls body" process. With years of improvements, nowadays deep learning models perform quite well at "Internet AI" tasks, probably because machines are naturally good at memorizing data. In fact, they have already surpassed human capabilities in many areas, such as recognizing objects in images [11] and lip-reading [18].

Recently, the focus is experiencing a shift to "embodied AI", active learning and planning within 3D environments. This is where AI agents learn and behave with both processes. However, it turns out that machines are not so good at active explorations and learning from experience, especially in our complicated and unordered real world. One reason could be that machines and programs are always somewhat inflexible, whereas our real world has so many variations.

2.3 Active Learning

One of the main differences between “Internet AI” and “embodied AI” is the learning pattern. In “Internet AI”, the whole training dataset is chunked into batches and fed to the model one by one on every epoch. The model cannot decide “what to learn” and “when to learn”; instead, it examines every sample in the training set in a passive manner. In “embodied AI” however, the model performs active learning. As is described in [21], the model is allowed to choose the data from which it learns. In embodied question answering [5], for instance, the agent may choose to visit some viewpoints several times while never arriving at other viewpoints.

2.4 Multi-modalities

The information in our world exists in multiple modalities: images, sounds, voices, smells, touches, etc. and we humans have learned to interpret and fuse all these sources for making decisions. In order for AI to carry out tasks in our world, it must also learn to deal with multi-modalities of info. Traditionally, “Internet AI” mainly deals with data in a single modality. Classification, detection and segmentation models only look at pixels; spam filter, sentiment analyzer and translation models only look at text. However, in “embodied AI”, information comes in different modalities, and the model should carry out different subtasks. In vision-language navigation tasks, including goal navigation [17] and instruction following [6], the agent receives a goal or a series of instructions specified by natural language; then it should self-navigate to the goal depending on the top-mounted camera. Note that the agent has no access to the map of the environment. In order to finish such tasks, the agent not only needs to understand the language but also actively explore the environment to gather visual information. Most importantly, it should learn to combine textual and visual information and finally map it into actions that lead the agent to its goal. Embodied question answering [5], or interactive question answering [9] can be seen as vision-language navigation followed by visual question answering [3]. Furthermore, in some recent work, embodied tasks with even more modalities of information are introduced, such as audio [4] and two-agent communications [12].

2.5 Common practices in embodied AI

Virtual environment. Considering safety issues, experiment cost, and reproducibility, embodied AI experiments are usually carried out in virtual 3D environments. From synthetic environments constructed by game engines such as SUNCG [20] and AI2THOR [24] in some early works, to nowadays photorealistic ones captured by special cameras such as Matterport3D [1] and Replica [8], an increasing number of 3D environments and simulators are being created. For simplicity, another common practice in experiment is to discretize the environment into blocks [9] or undirected graphs [17] and also discretize the possible actions of the agent accordingly.

Reinforcement learning. Most of the embodied AI tasks can be formulated into reinforcement learning problems where the agent maintains some policies, acts, transits to another stage and receives rewards from the environment. Particularly, imitation learning [22][23] is often used during training as it is easy to generate expert demonstrations (shortest path to the goal) from the environment. However, such following-shortest-path learning strategies can also result in poor generalization in unseen environments.

III. Method

As a large portion of this work is based on [17], we describe our methods by comparing with [17].

3.1 Characters

Both in this work and [17], four characters are participating in this whole learning problem: agent, navigation teacher, help-requesting teacher and advisor.

Agent. The AI agent can be considered as a robot with a top-mounted camera moving around in indoor environments. The agent runs two policies at each time step: navigation policy π_{nav} and help-requesting policy π_{help} .

The navigation policy π_{nav} decides which action to perform next. Specifically, it takes in the end goal, the current observation (RGB image), hidden states, last-step navigation action and ask action, etc., and outputs a probability distribution over a set of primitive actions:

$(forward, turn\_left, turn\_right, look\_up, look\_down, stop)$

In [17], π_{help} decides whether to signal the need for help based on the current situation. It also takes in the end goal, observation and etc., and outputs the probabilities of ask and $dont\_ask$. In our work, however, as the agent not only needs to decide **whether to ask**, but also **what to ask**, π_{help} outputs a probability distribution over a set of questions instead. (see the full list of questions in Fig 3)

Navigation teacher. Both the navigation teacher and the help-requesting teacher have full access to the environment information and are available during training. The navigation teacher can be seen as the optimal navigation policy π^*_{nav}, which provides optimal navigation actions for the agent. At each time step, it first finds the closest goal to the agent's current position. (Note that there can be more than one goal in a task, and the agent's arrival at any of the goals will be considered as a success. For example, for command "find a laptop in one of the bedrooms", there may be multiple laptops in different bedrooms). Then it calculates the shortest path to the nearest goal using Dijkstra Algorithm [7] (the environment is discretized into an undirected graph) and finally returns the optimal actions that move along the shortest path.

Help-requesting teacher. The help-requesting teacher can be seen as the optimal help-requesting policy π^*_{help}, which decides whether the agent should request for help based on its internal states and the environment. In [17], the help-requesting teacher is heuristic-driven, which decides that the agent should ask for help when any of the manually designed criterions are met. We modify some of the criterions to make them compatible with our Q&A settings. They are listed as follows:

(a) Deviate: the distance between the current position of the agent and its nearest goal is more than δ meters.

(b) Uncertain: the agent is "confused", defined as the KL Divergence between the agent's tentative navigation distribution p_{nav} and the uniform distribution is smaller than a threshold ϵ.

(c) Unmoved: The agent has remained at the same viewpoint for the last μ steps.

(d) Same Room: For the last μ' steps, the agent has remained in the same room and has not asked "Same Room" questions.

(e) Arrive: The agent is at a goal viewpoint or the highest probability action of the tentative navigation distribution is $stop$.

Again, as in our work, the help-requesting teacher should decide not only "**whether the agent should ask**", but also "**what the agent should ask**", we manually design some questions and did a simple mapping from criterions to questions denoted as $M1$ (see Fig 3).

Advisor. The advisor attends to the agent's requests or answers the agent's questions. Ideally, the advisor should be a human, who has no map of the environment and answers solely based on common sense. However, for the experiment, we assume the advisor to be another program which has complete information about the environment.

In [17], when the agent asks for help, the advisor would go and fetch optimal navigation actions for the next k steps from the navigation teacher π^*_{nav}, formulate these k actions into a fine-grained instruction in natural language, and prepend the instruction to the end goal (see Fig 2). At the next step the agent takes in the modified version of the end goal and thus gets the instruction.

In our work, the advisor answers the agent's questions by examining the agent's current position and the environment. Then it does another mapping $M2$ from answers to instructions which are also prepended to the end goal or replace the end goal (see Fig 3).

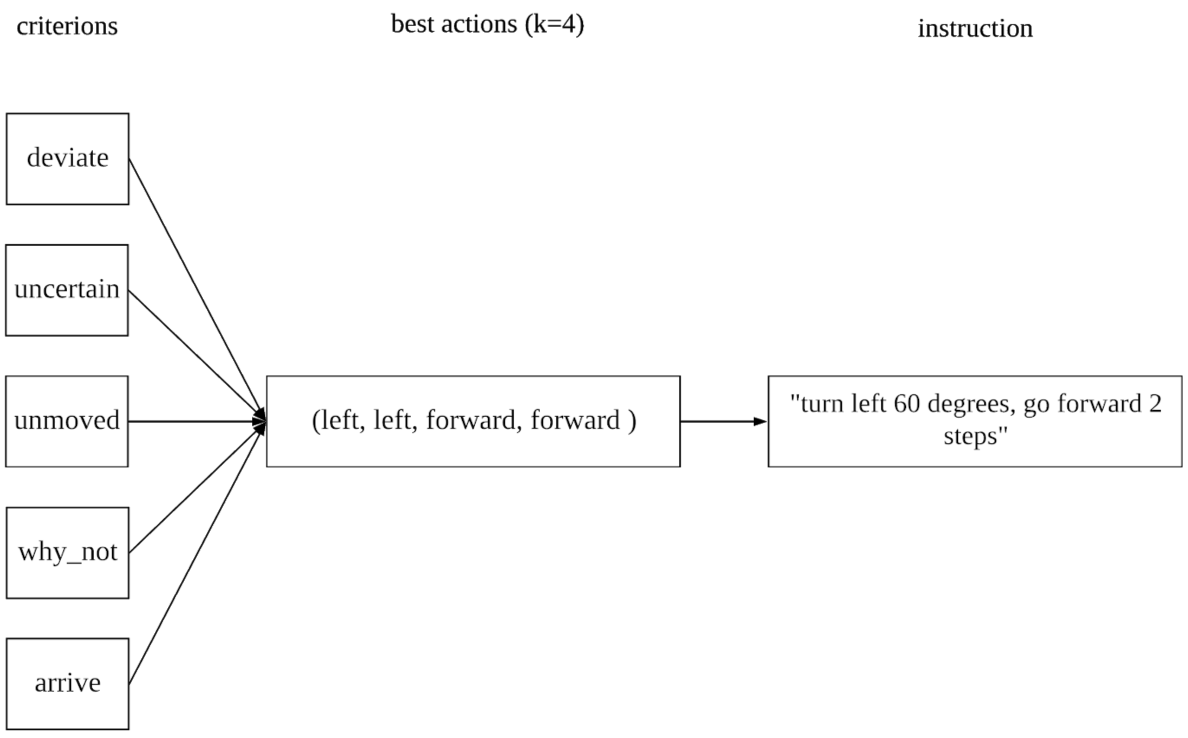

Fig 2: mapping from criterions to instructions in [17]

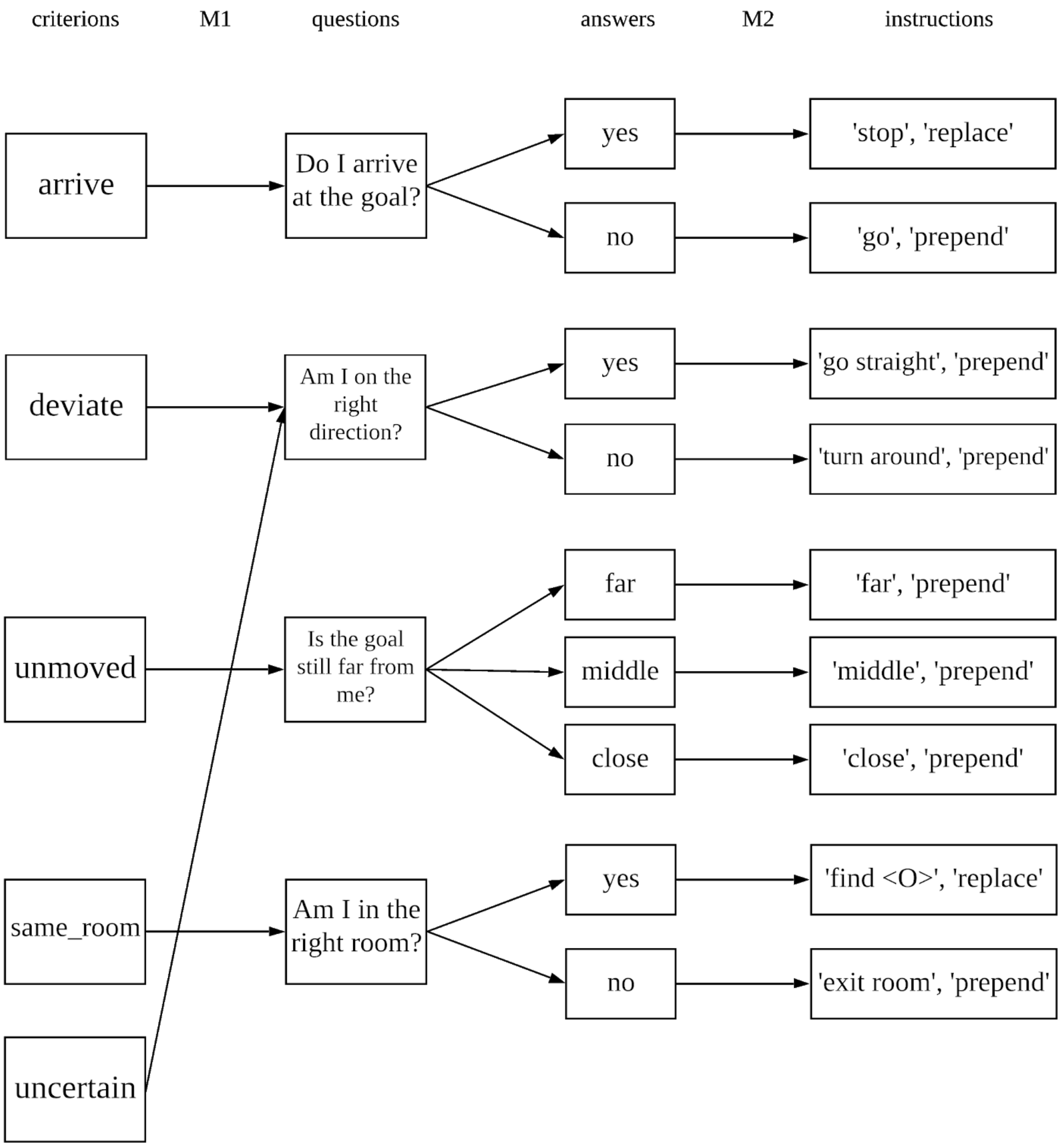

Fig 3: mapping from criterions to instructions in our work. Note that the order of the criterions matters: the help-requesting teacher π^*_{help} first examines the criterion at the top of this figure ("arrive"). If satisfied, π^*_{help} would output the corresponding question and ignore all the other criterions. Otherwise, π^*_{help} examines the next ("deviate") and continues.

3.2 Models

Most of the neural architecture in our work is the same as [17]. As illustrated in Fig 4, the navigation module is an encoder-decoder model with a multiplicative attention mechanism [14] and coverage modelling [25]. At step t,

1) The attended encoding of the end goal, image feature of the current observation o_t (by feeding the current viewpoint of RGB image into a ResNet-152 [10] pretrained on ImageNet [19]), last step hidden states h_{t-1}, last step navigation action a_{t-1}^{nav} and ask action a_{t-1}^{ask} are concatenated and fed into the decoder to predict a tentative navigation distribution $p_{t,tentative}^{nav}$.

2) The same input to the decoder, plus the ask budget b_t and the output from 1) are concatenated and fed to linear layers which generate the help-requesting distribution p_t^{ask}.

3) A new ask action a_t^{ask} is sampled from p_t^{ask}, and the end goal is modified by the advisor accordingly (if a_t^{ask} is not $dont\_ask$).

4) The modified goal is again fed to the encoder to get the new encoding, which is concatenated with the observation, hidden states, last-step navigation action and the current ask action and fed to the decoder to get the final navigation distribution $p_{t,final}^{nav}$.

See the detailed description of the process in [17].

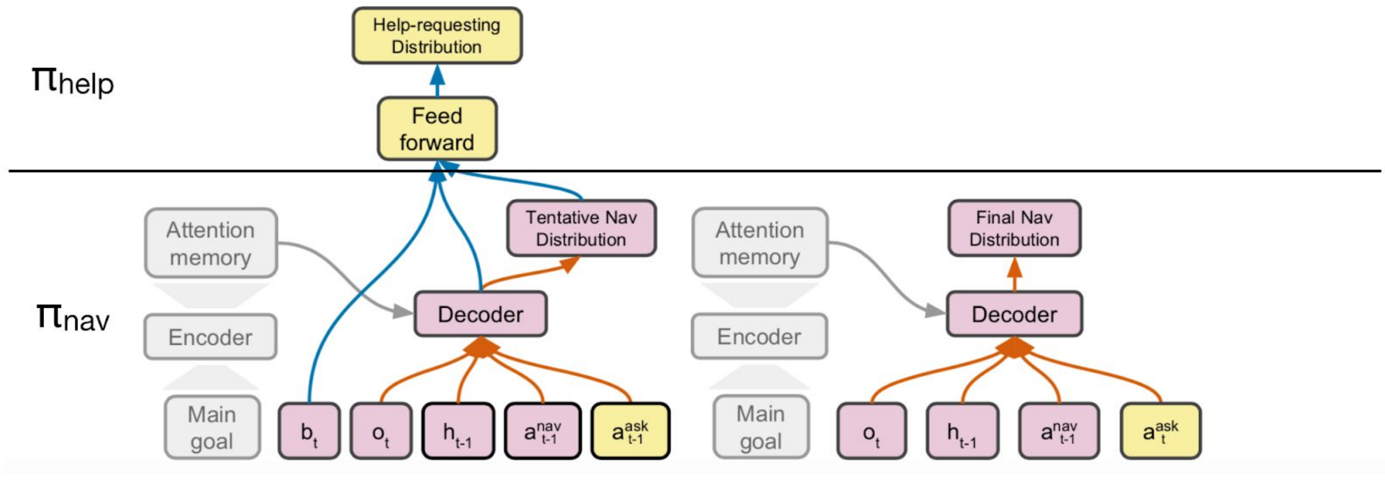

Fig 4: navigation and help-requesting modules

3.3 Training and Inference

We train the agent using imitation learning / supervised learning: the agent learns by mimicking the teachers' behaviors. Specifically, the optimal navigation action a^*_{nav} and ask action a^*_{ask} provided by the teachers π^*_{nav} and π^*_{help} are used for updating the agent's policies during training. The loss functions are calculated as:

$$L^{nav}_t = cross\_entropy(p^{nav}_{t,final}, a^*_{nav,t})$$

$$L^{ask}_t = cross\_entropy(p^{ask}_t, a^*_{ask,t})$$

Besides, in [17], the agent is also trained in a teacher-forcing manner: the agent's decisions are overwritten by optimal actions. For ask actions, $a^*_{ask,t}$ overwrites the sampled result from p^{ask}_t at every time step; for navigation actions, $a^*_{nav,t}$ overwrites the sampled result from $p^{nav}_{t,final}$ if the current time step t is within the next k step of the last ask. This teacher-forcing scheme ensures the coherence between the verbal instruction given to the agent and the agent's actual actions. In other words, the agent's interpretation loss of the verbal instruction is minimized. See Fig 5.

During inference, the help-requesting teacher is no longer available, and the navigation teacher only indirectly influences the agent. See Fig 6.

In our work, however, as the instructions are much coarser and less informative, we decide not to replace the agent's decisions by the optimums for both navigation and asking, for the following 2 reasons:

a) It is hard to define corresponding actions for some of the instructions like "far", "middle" and "close"

b) It can give the agent more freedom to explore and figure out its own interpretation of the instructions.

Experiments show that the removal of this overwriting design leads to better performance in our Q&A setting.

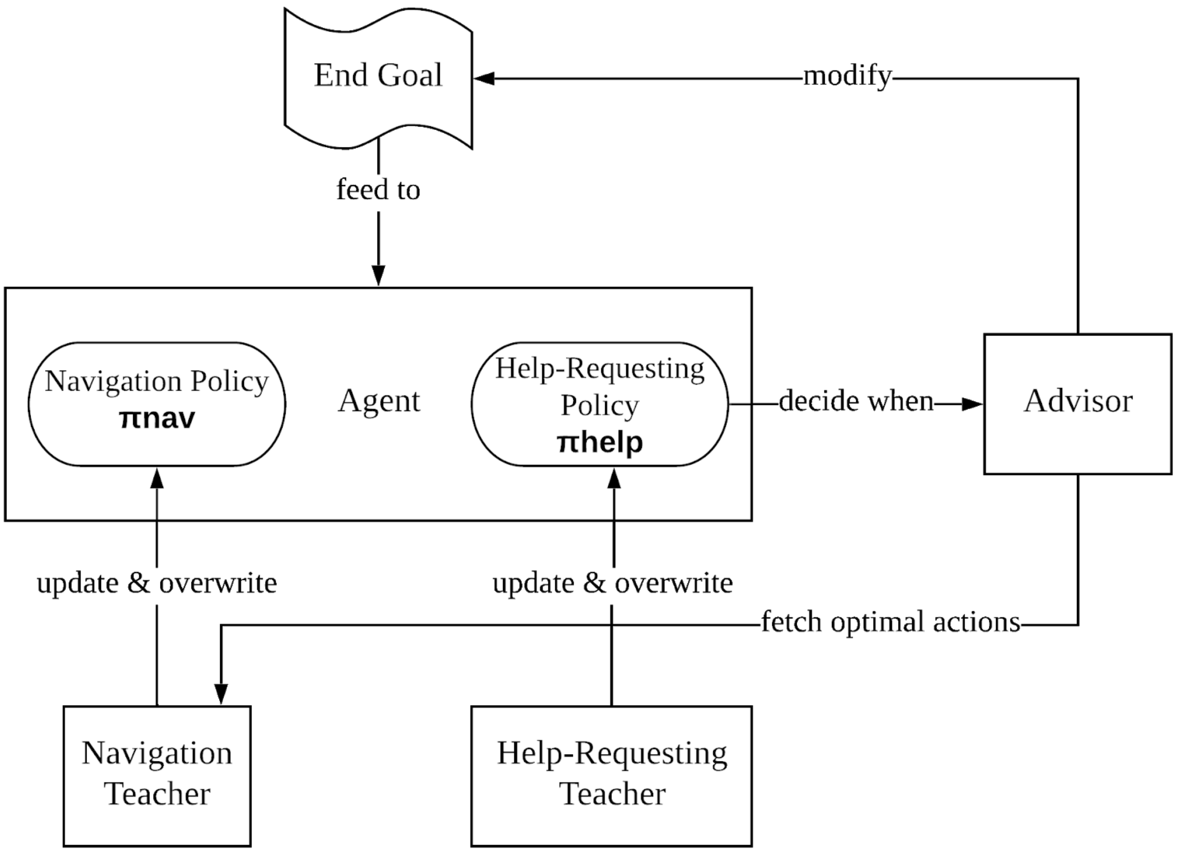

Fig 5: interactions among characters during training

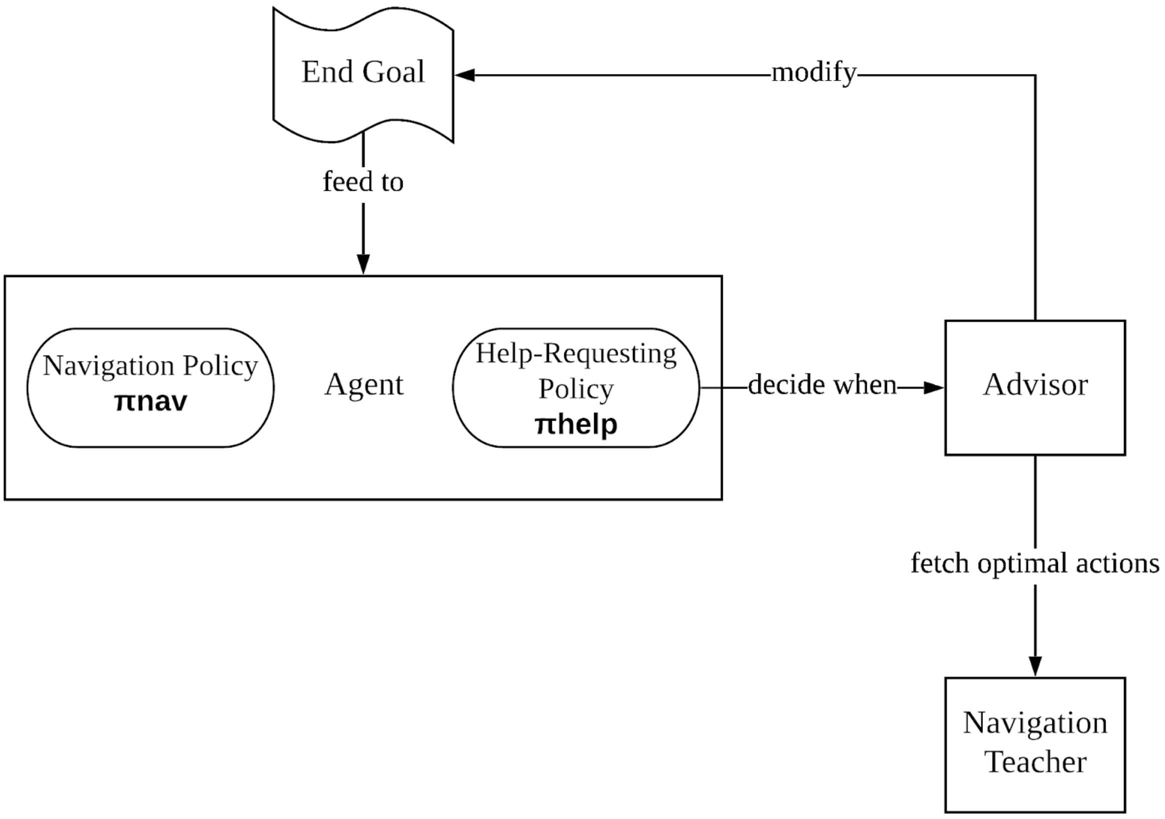

Fig 6: interactions among characters during inference

IV. Experiment

4.1 Setup

Configurations. All experiments are conducted on a single NVIDIA GTX 1080Ti with CUDA 10.0 in Ubuntu 16.04 LTS. Code written in Python 3.7.6 and Pytorch 1.0.2.

Evaluation metrics can be divided into 2 categories: navigational and asking. Navigational metrics measures how the agent performs in navigating to goals, which are also the main metrics of this work. Definitions are listed in Table 1 below.

Metric Name	Description
$success\_rate$	fraction of test set on which the agent stops at a point within d meters from any of the goals
$oracle\_success\_rate$	fraction of test set on which the any point on the agent's trajectory falls within d meters from any of the goals
$room\_success\_rate$	fraction of test set on which the agent stops at any room containing a goal
nav_error	average distance from the point where the agent stops to its closest goal
$oracle\_nav\_error$	average of the smallest distances from any point on the agent's trajectory to any goal.
$length$	average distance between where the agent starts and stops
$steps$	average number of steps the agent takes on each trajectory

Table 1: navigational metrics

Asking metrics measures the quantities and qualities of the questions the agent asks. Definitions are listed in Table 2 below.

Metric Name	Description
$queries\_per\_ep$	average number of questions asked per episode (per trajectory)
$A/P/R/F$	average accuracy, precision, recall and F1 score (macro) of the question types asked by the agent comparing to the teacher
$bad\_questions\_per\_ep$	average number of bad questions asked per episode

Table 2: asking metrics

In the third asking metric $bad\_questions\_per\_ep$, we count the number of "bad" questions. A question is considered as "bad" when:

a) The agent asks a second or more "am I on the right direction" questions in the same location and with the same heading.

b) The agent asks a second or more "is the goal still far from me" questions without much change in the distance to the goal (less than 3 meters).

c) The agent asks a second or more "am I the right room" questions in the same room.

d) The agent asks "do I arrive at the goal" question when still far from the goal (more than 5 meters).

We design the $bad\_questions\_per\_ep$ metric in addition to $A/P/R/F$, as the latter is not necessarily a proper measurement of the question qualities in our Q&A setting. It only measures how good the agent is at imitating the teacher's behaviors. Although we manually create some criterions and mappings for the teacher, the resulting policy π^*_{help} may not be "optimal". In fact, it is tough to define which question is the "optimal" question at every step. Maybe one question is better than another, i.e., can provide more information in certain situations, but often it is not the only acceptable one. As a result, a good imitator asks high-quality questions, but a lousy imitator does not always ask useless questions. Therefore, we need some manually designed criterions to judge the quality of a question. In experiments, however, we still train the agent by imitating the teacher, and only record but do not punish the agent for asking "bad" questions.

Hyperparameters. Comparing to the hyperparameters in [17], we have one more parameter $same\_room\_threshold$, which corresponds to $\mu^{'}$ in the "same_room" criterion of the help-requesting teacher. Parameter k, which indicates the number of actions suggested by a subgoal, is rendered invalid thus removed in our Q&A setting. δ, which indicates the deviate threshold, has different context from the original work. To encourage the agent to ask various questions instead of solely relying on a certain type, we lower the uncertainty threshold ϵ to 0.9 and the unmoved threshold μ to 7, so that the probabilities of the help-requesting teacher meeting different criterions are more balanced. Also, as the answers provided contain little information, we increase the help-requesting ratio τ to 0.8, which grants the agent a higher budget for the number of questions it may ask. For all the other unmentioned hyperparameters, they remain the same as [17].

4.2 Results

We compare our results with [17] in terms of $success\_rate$, $room\_success\_rate$ and nav_error. Results are shown separately for the 2 portions of the test set: "seen" and "unseen". In "seen", the environments have been exposed to the agent during training, but end goals are altered. In "unseen", environments are also new to the agent, i.e. they do not appear in the training set. The result in [17] is referred to as "vnla". Baseline "none", where the agent does not ask any questions, is also added here for comparison.

	$success\_rate$	$room\_success\_rate$	nav_error
		seen	
vnla	52.26	65.42	3.42
none	28.39	48.97	6.29
ours	42.27	58.76	4.40
		unseen	
vnla	34.95	44.85	5.61
none	6.36	14.34	11.30
ours	21.02	29.87	7.44

Table 3: main results compared with [17]

As is expected, the performance of our agent is worse than [17], as it is much harder for the agent to learn from our short and vague answers compared to the fine-grained instructions. However, our agent still performs much better than “none”, especially in unseen environments. We can infer that the agent has effectively learned to ask questions and interpret answers, which provides much help in fulfilling the navigation task.

Results for all other metrics mentioned above are further released in Table 4 & 5.

	$success\_rate$	$oracle\_success\_rate$	$room\_success\_rate$	nav_error	$oracle\_nav\_error$	$length$	$steps$
seen	42.27	51.87	58.76	4.40	3.57	11.32	15.31
unseen	21.02	27.60	29.87	7.44	6.62	8.01	15.79

Table 4: navigational metrics result

	$queries\_per\_eps$	$A/P/R/F$	$bad\_questions\_per\_episode$
seen	4.0	0.72 / 0.72 / 0.46 / 0.49	0.2
unseen	7.1	0.67 / 0.64 / 0.45 / 0.45	0.7

Table 5: asking metrics result

Finally, we plot out the number of questions asked at different steps in Fig 7. Although we have tuned the thresholds to balance question types, the agent still heavily relies on "direction" questions, especially in the first few steps of an episode. As the agent learns by imitating the teacher's behavior, if we look at the mapping from asking criterions to questions $M1$, it can be inferred that the agent is highly possible to get "confused" at the start of an episode. More "distance" questions appear in the middle of an episode, which means that sometimes the agent may stop moving halfway. Unsurprisingly, "arrive" and "room" questions are seldom seen, as their corresponding criterions are either strict or appear at the lower rank of the priority list.

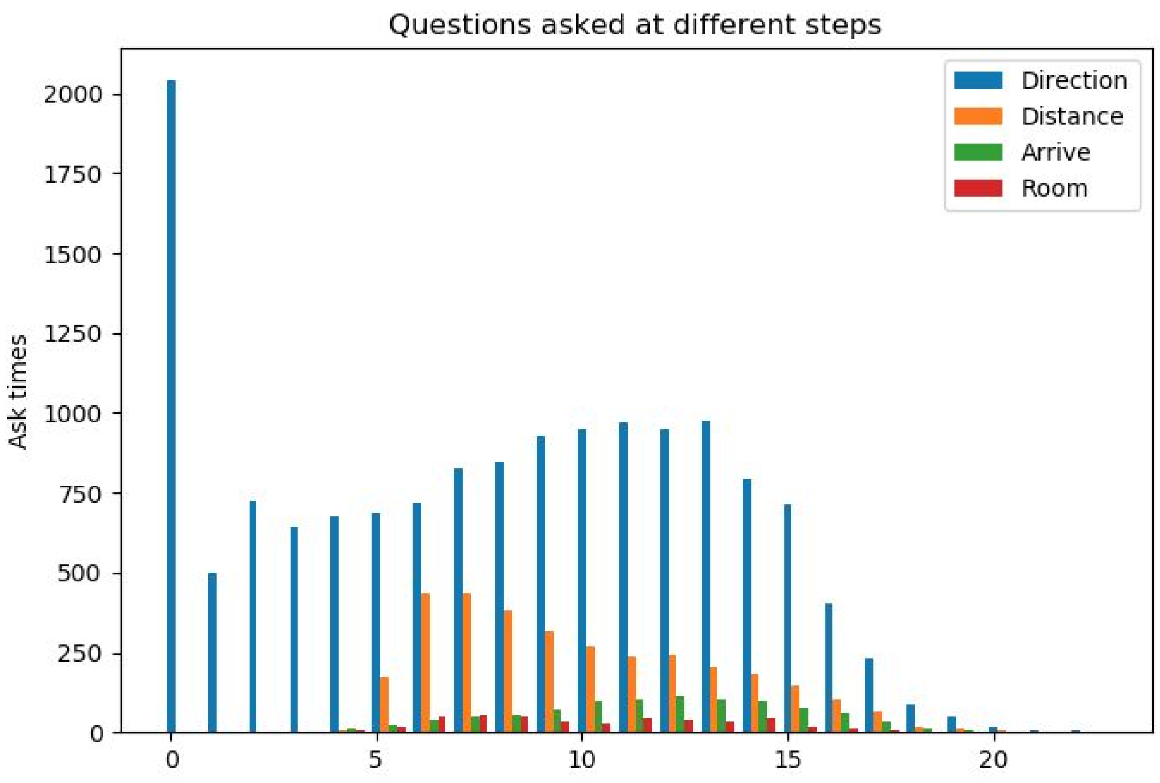

Fig 7: questions asked at different steps

V. Conclusion and Future Work

In this work, we trained an AI agent via imitating two teachers. The agent has learned to perceive instructions, navigate in indoor environments, and most importantly, actively ask questions to help its navigational decisions. However, there are a few points that can be further improved on.

Firstly, the agent does poorly in generalizing to unseen environments, as is shown in the experiment result section, there is a huge gap between the agent's performances in seen and unseen environments. This is an expected result, as we only trained the agent via imitation learning. To remedy the problem, we may define a reward function considering the distance between the agent's position and the goal, and fine-tune the agent using REINFORCE [16] or A3C [26] algorithms after the imitation learning stage.

Secondly, questions and answers are all restricted to a small-size pre-defined set. This is because our task, the goal-driven navigation, is rather straightforward and few questions can contribute to better performance. To allow the agent to ask more diversified types of questions, it should be put into more complex situations, especially those where verbal messages can deliver much information and are essential for the success of the task.

VI. References

[1] *Matterport3D: Learning from RGB-D Data in Indoor Environments*, niessner.github.io/Matterport/.

[2] Aghaebrahimian, Ahmad, and Filip Jurčíček. “Open-Domain Factoid Question Answering via Knowledge Graph Search.” *Proceedings of the Workshop on Human-Computer Question Answering*, 2016, doi:10.18653/v1/w16-0104.

[3] Agrawal, Aishwarya, et al. “VQA: Visual Question Answering.” *International Journal of Computer Vision*, vol. 123, no. 1, 2016, pp. 4–31., doi:10.1007/s11263-016-0966-6.

[4] Changan, Chen, et al.(2019). “Audio-Visual Embodied Navigation.” arXiv:1912.11474.

[5] Das, Abhishek, et al. “Embodied Question Answering.” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, doi:10.1109/cvpr.2018.00008.

[6] Devo, Alessandro, et al. “Deep Reinforcement Learning for Instruction Following Visual Navigation in 3D Maze-Like Environments.” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, 2020, pp. 1175–1182., doi:10.1109/lra.2020.2965857.

[7] Dijkstra, E. W. “A Note on Two Problems in Connexion with Graphs.” *Numerische Mathematik*, vol. 1, no. 1, 1959, pp. 269–271., doi:10.1007/bf01386390.

[8] Facebookresearch. “Facebookresearch/Replica-Dataset.” *GitHub*, 19 July 2019, github.com/facebookresearch/Replica-Dataset.

[9] Gordon, Daniel, et al. “IQA: Visual Question Answering in Interactive Environments.” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, doi:10.1109/cvpr.2018.00430.

[10] He, Kaiming, et al. “Deep Residual Learning for Image Recognition.” *2016 IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), 2016, doi:10.1109/cvpr.2016.90.

[11] He, Kaiming, et al. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.” *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, doi:10.1109/iccv.2015.123.

[12] Jain, Unnat, et al. “Two Body Problem: Collaborative Visual Task Completion.” *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, doi:10.1109/cvpr.2019.00685.

[13] Lukovnikov, Denis, et al. “Neural Network-Based Question Answering over Knowledge Graphs on Word and Character Level.” *Proceedings of the 26th International Conference on World Wide Web - WWW '17*, 2017, doi:10.1145/3038912.3052675.

[14] Luong, Thang, et al. “Effective Approaches to Attention-Based Neural Machine Translation.” *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, doi:10.18653/v1/d15-1166.

[15] McNerney, Samuel. “A Brief Guide to Embodied Cognition: Why You Are Not Your Brain.” *Scientific American Blog Network*, Scientific American, 4 Nov. 2011, blogs.scientificamerican.com/guest-blog/a-brief-guide-to-embodied-cognition-why-you-are-not-your-brain/.

[16] Mnih, et al. “Asynchronous Methods for Deep Reinforcement Learning.” arXiv:1602.01783.

[17] Nguyen, Khanh, et al. “Vision-Based Navigation With Language-Based Assistance via Imitation Learning With Indirect Intervention.” *2019 IEEE/CVF Conference on*

Computer Vision and Pattern Recognition (CVPR), 2019, doi:10.1109/cvpr.2019.01281.

[18] Petridis, Stavros, et al. “End-To-End Multi-View Lipreading.” *Procedings of the British Machine Vision Conference 2017*, 2017, doi:10.5244/c.31.161.

[19] Russakovsky, Olga, et al. “ImageNet Large Scale Visual Recognition Challenge.” *International Journal of Computer Vision*, vol. 115, no. 3, 2015, pp. 211–252., doi:10.1007/s11263-015-0816-y.

[20] “Semantic Scene Completion.” *Princeton University*, The Trustees of Princeton University, sscnet.cs.princeton.edu/.

[21] Settles, Burr. “Active Learning.” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, 2012, pp. 1–114., doi:10.2200/s00429ed1v01y201207aim018.

[22] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668, 2010

[23] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no- regret online learning. In *Proceedings of the fourteenth inter- national conference on artificial intelligence and statistics*, pages 627–635, 2011

[24] “THOR.” *AI2*, ai2thor.allenai.org/.

[25] Tu, Zhaopeng, et al. “Modeling Coverage for Neural Machine Translation.” *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, doi:10.18653/v1/p16-1008.

[26] Williams, Ronald J. “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning.” *Reinforcement Learning*, 1992, pp. 5–32., doi:10.1007/978-1-4615-3618-5_2.