PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

BRIDGING THE VISUAL SEMANTIC GAP IN VLN VIA SEMANTICALLY RICHER INSTRUCTIONS

JOAQUÍN OSSANDÓN STANKE

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Advisor:
ÁLVARO SOTO ARRIAZA

Santiago de Chile, July 2022

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

BRIDGING THE VISUAL SEMANTIC GAP IN VLN VIA SEMANTICALLY RICHER INSTRUCTIONS

JOAQUÍN OSSANDÓN STANKE

Members of the Committee:

ÁLVARO SOTO ARRIAZA DocuSigned by: Alvaro Soto DA096D318D8B4F2...

HANS LÖBEL DÍAZ DocuSigned by: Hans Löbel 8659C4EF6B9449D...

FELIPE BRAVO MÁRQUEZ DocuSigned by: Felipe Bravo 36D4C8B7989E468...

IGNACIO VARGAS CUCURELLA DocuSigned by: Ignacio Vargas F1FD14CD7E7644C...

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Santiago de Chile, July 2022

For my family and friends, who
patiently listened to all my ideas
over these two years

ACKNOWLEDGEMENTS

I want to start thanking my advisor Álvaro Soto for accepting me, trusting and supporting me throughout this journey. Thank you for believing in my abilities, my enthusiasm and my desire to learn about this beautiful topic from the beginning.

The IA Lab group was essential to maintain motivation and learning from different areas, sharing almost every week. Thank you Gabriel, Francisca, Jorge, Juan Pablo, Patricio and Felipe for your good vibes. Special thanks to Benjamin, for so many ideas and collaboration in this work.

Thanks to my parents, Pablo and Sylvia, for always supporting me with my decisions, encouraging to always bring out the best in me. To my sisters, Consuelo and Natalia, for bearing with me and always being there. To all my family. Thank you.

This journey would not have been the same without the people around me. Special mention to Catalina, Camila, Sebastian, Felipe and Alejandro, who patiently listened to all my ideas, cheering me up me whenever I was discouraged.

To all the people who were with me these two years, thank you. I was fortunate to get to know all of you.

This is just the beginning!

This work received financial support from the Chilean National Agency for Research and Development (ANID) through *Beca de Magíster Nacional* N° 22210030.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iv

LIST OF FIGURES vii

LIST OF TABLES ix

ABSTRACT x

RESUMEN xi

1. INTRODUCTION 1

2. THE VISUAL AND LANGUAGE NAVIGATION TASK 6
   2.1. Description 6
   2.2. Comparison metrics 7
   2.3. New problems 7

3. RELATED WORK 10
   3.1. Auxiliary tasks 10
   3.2. Navigation and exploration 11
   3.3. Curriculum learning, pre-training and data augmentation 11
   3.4. Leaderboard 12

4. RESEARCH QUESTIONS AND HYPOTHESES 14

5. TESTING THE RELEVANCE OF VISUAL INFORMATION IN VLN 15
   5.1. Experiments in the visual area 15
   5.2. Experiments in the language area 16
   5.3. Ablation studies results 18

6. SEMANTICALLY RICHER INSTRUCTIONS 19
   6.1. Objects 20
      6.1.1. Metadata parser 21

6.2. Objects instructions 23
6.3. Crafted instructions 25

7. RESULTS AND DISCUSSION 28
7.1. Qualitative analysis of generated instructions 28
7.2. Evaluation of the Regretful Agent pre-trained with generated instructions . 30
7.3. Assesment of human wayfinding 35
7.3.1. Panoramic view 36
7.3.2. Rotate 36
7.3.3. Move to navigable node 36
7.3.4. Stop and retrieve metrics 37
7.3.5. Collected data 40

8. CONCLUSIONS 42

9. FUTURE WORK 44

REFERENCES 45

APPENDIX 50
A. Matterport3DMetadata parser 51
A.1. Regions 51
A.2. Objects 51
B. Crafted and Generated Instructions 54
B.1. Crafted instructions 54
B.2. Generated instruction from baseline module with auxiliary tasks 57
C. Pre-training the Regretful Agent with generated instructions 57

LIST OF FIGURES

1.1 Examples of tasks involving visual and text modalities. 2

2.1 Objects and viewpoints visualization through `360-visualization` scripts. 7

2.2 New problems based on the Visual-and-language Navigation Task. 9

5.1 Ablation studies setup. 15

6.1 Speaker language model architecture with the proposed auxiliary tasks: objects auxiliary task and crafted auxiliary task. 20

6.2 Objects and nodes retrieved with `360-visualization` for a specific node on a path. 21

6.3 Scan `Z6MFQCViBuw` raw metadata information. 22

6.4 Sampled regions of scan `Z6MFQCViBuw` parsed from raw metadata. 22

6.5 Sampled objects of scan `Z6MFQCViBuw` parsed from raw metadata. 23

6.6 Reachable viewpoints information of a sampled viewpoint parsed from raw metadata. 23

6.7 Visible objects information of a sampled viewpoint parsed from raw metadata. 24

6.8 Objects instruction generated for the path and the resulting negative log loss applied to the generated instruction on training. 24

6.9 Agent orientation mapping on a 360° image. 25

6.10 Crafted instruction example. 26

6.11 Crafted instruction generated for the path and the resulting negative log loss applied to generated instruction on training. 27

7.1 Comparison of human instruction with instructions generated using the Speaker base model, instructions generated using the Speaker with auxiliary tasks and our crafted instructions. 29

7.2 Success rate of Regretful-Agent pre-trained with instructions generated with Speaker trained with the two best configurations of λ and β on unseen environments. 33

7.3 Landing of the instruction follower platform. 36

7.4 Available actions on the instruction follower platform. 37

7.5 Sequence after rotating left on instruction follower platform 37

7.6 Sequence after moving to node "2" on instruction follower platform 38

7.7 Successful navigation on instruction follower platform. 38

7.8 JSON output sample after finishing route on instruction follower platform. . . 39

A.1 Environment descriptions used to map regions letter labels. 51

A.2 Objects names used to map objects labels indexes. 52

A.3 Distance between a node and an object calculated relative to the current node. 53

B.1 First crafted instructions example. 58

B.2 Second crafted instructions example. 59

B.3 Baseline module with auxiliary tasks output example. 60

C.1 Graph of success rate weighted by path length through epochs. 61

C.2 Graph of distance from goal through epochs. 61

LIST OF TABLES

2.1 Description of metrics used in this work. 8

3.1 Comparison of the different models solving the Room-to-Room task, in unseen test set using Single Run. 13

5.1 Visual ablation study on Self-Monitoring and Regretful-Agent, **seen** environments with Single Run. 17

5.2 Visual ablation study on Self-Monitoring and Regretful-Agent, **unseen** environments with Single Run. 17

5.3 Language ablation study on Self-Monitoring and Regretful-Agent, **unseen** environments with Single Run. Best success rate is in bold. 17

7.1 Regretful-Agent pre-training with instructions generated from different sources, evaluated on **seen** environments with Single Run and original human instructions. Best success rate is in bold. 31

7.2 Regretful-Agent pre-training with instructions generated from different sources, evaluated on **unseen** environments with Single Run and original human instructions. Best success rate is in bold. 32

7.3 Summary table of the performance of the Regretful-Agent pre-trained with instructions generated from different sources, evaluated on **unseen** environments with Single Run and original human instructions. 34

7.4 Average metrics of the first 15 observations for each follower when navigating in unseen environments. 40

ABSTRACT

The Visual-and-Language Navigation (VLN) task requires understanding a textual instruction to navigate a natural indoor environment using only visual information. While this is a trivial task for most humans, it is still an open problem for AI models. In this work, we hypothesize that poor use of the visual information available is at the core of the low performance of current models. To support this hypothesis, we provide experimental evidence showing that state-of-art models are not severely affected when they receive just limited or even no visual data, indicating a strong overfitting to the textual instructions.

To encourage a more suitable use of the visual information, we propose a new data augmentation method that fosters the inclusion of more explicit visual information in the generation of textual navigational instructions. Our main intuition is that current VLN datasets include textual instructions that are intended to inform an expert navigator, such as a human, but not a beginner visual navigational agent, such as a randomly initialized deep learning model. Specifically, to bridge the visual semantic gap of current VLN datasets, we take advantage of metadata available for the Matterport3D dataset that, among others, includes information about object labels that are present in the scenes.

Training a state-of-the-art model with the new set of instructions increase its performance by 8% in terms of success rate on unseen environments, while testing these new instructions on humans outperforms available synthetic instructions, demonstrating the advantages of the proposed data augmentation method.

Keywords: visual-and-language, navigation, VLN, vision, cognitive robotics.

RESUMEN

La tarea de Visual-and-Language Navigation (VLN) requiere entender complejas instrucciones de texto en lenguaje natural y navegar en un ambiente natural interior usando únicamente información visual. Mientras es una tarea trivial para el humano, sigue siendo un problema abierto para los modelos de inteligencia artificial. En este trabajo, planteamos como hipótesis que el mal uso de la información visual disponible es la razón principal del bajo rendimiento de los modelos actuales. Para apoyar esta hipótesis, presentamos evidencia experimental mostrando que modelos del estado del arte no son totalmente afectados cuando reciben limitada o incluso nula información visual, indicando un fuerte *overfitting* al texto de las instrucciones.

Para fomentar un uso más adecuado de la información visual, proponemos un nuevo método de aumento de datos que fomenta la inclusión de información visual más explícita en la generación de instrucciones de navegación textuales. Nuestra intuición principal es que los conjuntos de datos actuales incluyen instrucciones textuales que tienen como objetivo informar a un navegante experto, como un ser humano, pero no a un agente de navegación visual principiante, como un modelo de *deep learning* inicializado aleatoriamente. Específicamente, para cerrar la brecha semántica visual de los conjuntos de datos actuales, aprovechamos los metadatos disponibles para el conjunto de datos Matterport3D que, entre otros, incluye información sobre etiquetas de objetos que están presentes en las escenas.

Entrenando un modelo actual con el nuevo conjunto de instrucciones generado aumenta su rendimiento en un 8% en cuanto a tasa de éxito en entornos desconocidos, mientras que probar estas nuevas instrucciones en humanos supera a las instrucciones sintéticas disponibles, lo que demuestra las ventajas de la propuesta de aumento de datos.

Palabras Clave: visión, lenguaje, navegación, VLN, robótica cognitiva.

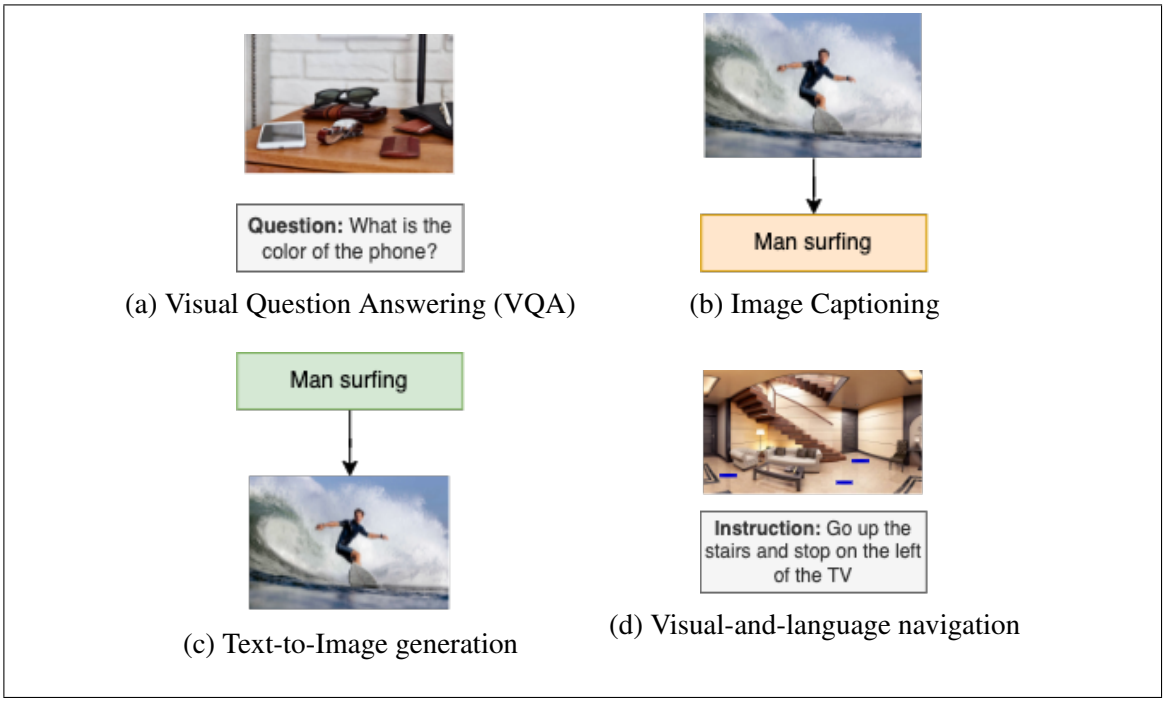
1. INTRODUCTION

Service robots have received increased attention in recent years, with broad applications in many fields, such as healthcare, education, entertainment, logistics or domestic tasks (García, Strüber, Brugali, Berger, & Pelliccione, 2020). A service robot is a "robot that performs useful tasks for humans or equipment excluding industrial automation applications" (ISO, 2012). We are seeing service robots more and more frequently at home. In fact, service robotics market was valued at USD $23577.1 million in 2020 and is expected to grow at a compound annual rate of 44.9% by 2026 (MordorIntelligence, 2021).

The performance of domestic tasks is one of the major problems facing advanced societies today. The development of equipment capable of reducing the workload in home has been going on for more than fifty years with the invasion of household appliances, many of which are indispensable in every home. Among these appliances, there are autonomous robots defined as intelligent machines capable of performing tasks in the world by themselves, without explicit human control (Bekey, 2005).

Some of these autonomous robots involve vision and language, basing their operation on two subareas of AI: Computer Vision and Natural Language Processing. Computer Vision is about describing the world that we see in one or more images. This task is used in a wide variety of real-world applications, such as Optical Character Recognition, for recognizing text, Object or Face Detection or Image Transformation (Szeliski, 2011). Natural Language Processing, on the other hand, is a technique that uses algorithms to analyze or transform textual data, used for Question-Answering, Machine Translation, Information Extraction, and more (Liddy, 2001).

Over these research subareas useful models had been developed, making substantial advances, even sometimes surpassing human performance. However, there are complex tasks that include both vision and language, where some of them have not yet been achieved good performance. Examples are Visual-Question Answering, where the goal is to answer a question having an image (Antol et al., 2015), Image captioning, where



(a) Visual Question Answering (VQA)

(b) Image Captioning

(c) Text-to-Image generation

(d) Visual-and-language navigation

Figure 1.1. Examples of tasks involving visual and text modalities.

agents generate descriptions of an image (Hossain, Sohel, Shiratuddin, & Laga, 2019), Image generation, where models transform text into a visual representation, and Visual-and-Language Navigation, where an agent is trained to navigate in real environments following natural language instructions (Anderson et al., 2018). Examples are shown on Image 1.1.

We focus this work on the Vision-and-language Navigation (VLN) task, where a robot receives an instruction in natural language and has to navigate to a goal position in known and unknown environments using visual information. This has been a very attractive research topic in recent years (Fried et al., 2018; Wang, Wang, Shu, Liang, & Shen, 2020; Tan, Yu, & Bansal, 2019; Manterola, 2021; Zhu, Zhu, Chang, & Liang, 2020; Majumdar et al., 2020). The Visual-and-Language Navigation (VLN) task proposes that an agent can follow an instruction such as "*Go up the stairs, turn right, and stop right at the left of the table*", and use it to navigate a natural indoor environment from a starting to a goal position using only visual information. In spite of current advances in AI, this task, that

results trivial for most humans, it still out of reach for autonomous robots. As an example, under current benchmarks (Wu, Chang, & Li, 2021), state-of-the-art AI models based on Deep Learning (DL) do not reach the intended goal position more than 65% of the time (Liu et al., 2021).

There are several reasons that can help to explain the low performance of current models to face the VLN task (Wu et al., 2021). Among them, we believe that lack of a proper visual understanding of the environment is a key factor. In effect, humans actively use relevant views of the environment to identify visual semantic information such as navigational cues, objects, scenes, or other situations, however, current AI models focus their operation on identifying relevant correlations between the textual instructions and visual data present in the training set (Marcus, 2018). As a consequence, current VLN models exhibit limited generalization capabilities, leading to a large drop in performance when they are tested in unseen environments (Fried et al., 2018; Tan et al., 2019; Wu et al., 2021).

There is abundant experimental evidence indicating that current DL based models operate as associative memory engines triggered by superficial data correlations (Jo & Bengio, 2017; Arpit et al., 2017; Belkin, Hsu, & Mitra, 2018), fostering the detection of direct stimulus-response associations. Indeed, given enough parameters, DL models are able to memorize arbitrary noisy data (Zhang, Bengio, Hardt, Recht, & Vinyals, 2017). In the case of VLN, this problem leads to a poor use of the visual information. As a consequence, instead of unveiling the richness of the visual world, DL models limit their operation to memorize low level correlations between textual and visual data. Even worse, in several cases, models ignore completely the visual information, learning a direct mapping between the textual instructions and robot action.

To support the previous observation, as a first contribution of this work, we provide experimental evidence indicating that current VLN models do not make a suitable use of the visual information available about the environment. Specifically, we demonstrate that when we provide to the model just limited or even no visual data, the model exhibits just

a slight drop in performance, showing that their operation is heavily biased to the use of textual instructions.

The previous observation motivates our main research question: how can we contribute to improve the use visual information in VLN models?. While the answer to this question is manifold, in this work we focus our contribution on the generation of more suitable training data. Specifically, we believe that a relevant problem of current VLN datasets is that, during their generation, the humans providing the textual instructions assume that they are intended for an expert navigator, as an example, another human. We believe that this scheme leads to the generation of high level textual instructions, where it is hardly complex to extract meaningful visual cues to inform a beginner visual navigational agent, such as a randomly initialized DL model. As a consequence, we believe that the data generation for a beginner should include a more detailed description of the visual world around the agent.

To bridge the visual semantic gap of current VLN datasets, we present a new data augmentation method that fosters the inclusion of more explicit visual information in the generation of textual navigational instructions. To do this, we resort to object labels present in the metadata available for the Matterport3D dataset[1] that we refer here as Matterport3DMeta. Using this data, we propose new semantically richer natural language instructions for the Room-to-Room (R2R) dataset (Anderson et al., 2018) that are generated with an improved version of the Speaker-Follower model (Fried et al., 2018). Specifically, we use scene objects and crafted instructions created with a set of rules that we encode to feed a set of auxiliary visual tasks. As a main finding, the resulting navigational instructions provide a significant boost in the performance of current VLN models when they are tested in previously unseeing environments.

As a further contribution, we make available the semantically enriched dataset generated in this work as well as a set of software tools to generate further data. These tools incorporate modules to access scene nodes in the Matterport3D dataset (Chang et al., 2017)

[1] https://github.com/niessner/Matterport/tree/master/metadata

that include information about relevant objects, their position, size, distance, heading, and elevation. We believe that this is a powerful starting point to use scene metadata to create semantically richer visual navigational instructions.

This work is organized as follows. On Section 2 we describe the VLN task and presenting the new problems that have arisen based on it. Section 3 reviews and categorize relevant previous work. Section 5 presents an experimental setup to highlight the limitations of current VLN models to use visual information. Afterwards, Section 6 describes the objects retrieval and the construction of our visual semantically richer instructions for the VLN task. Then, Section 7 present and discuss the results obtained from testing these generated instructions on navigation agents and humans. Finally, Sections 8 and 9 present our conclusions and future research avenues.

2. THE VISUAL AND LANGUAGE NAVIGATION TASK

During the last decade several studies have been related to the VLN task, however, the visual aspect was discarded due to the lack of real images in the proposed problems (Anderson et al., 2018). In 2017, the Matterport3D dataset (Chang et al., 2017) was introduced, containing RGB-D building scale scenes of 90 different home environments. Later that year, a new navigation problem is proposed: Room-to-Room (R2R) (Anderson et al., 2018), the first dataset for the Visual-and-Language Navigation task (VLN) on real 3D environments, introducing a Matterport3D based simulator, which simulates its environments with the possibility of navigate trough them. In R2R, 90 different environments from Matterport3D have been divided into training and validation (seen and unseen) splits. There are a total of 7,189 distinct paths (starting point, target point), with 3 distinct human instructions for each, totaling 21,567 navigation instructions with an average of 29 words (Anderson et al., 2018).

We construct over Matterport3DMeta a set of tools for getting objects and navigable nodes for each view, as shown in Figure 2.1. We describe this set of tools on Section 6.1.

2.1. Description

The task of R2R for an agent is to follow natural language instructions from an initial position to a target position through navigation in a real environment, simulated by Matterport3D Simulator (Anderson et al., 2018). At the beginning of each episode an instruction $\overline{x} = \langle x_1, x_2, .., x_L \rangle$ is given, where L is the instruction length and x_i a word token. The agent observes an RGB image v_0 depending on an initial 3D position, heading ψ_0 and elevation θ_0, resulting in a world state $s_0 = \langle v_0, \psi_0, \theta_0 \rangle$. The agent must execute a sequence of actions $\langle s_0, a_0, s_1, a_1, .., s_T, a_T \rangle$ where each action a_t leads to a new state $s_{t+1} = \langle v_{t+1}, \psi_{t+1}, \theta_{t+1} \rangle$ and generates a new visual panoramic view v_{t+1}. It is important to note that actions are given by the simulator, which are limited according to the node where the agent is located. The episode ends when the agent selects the <STOP>

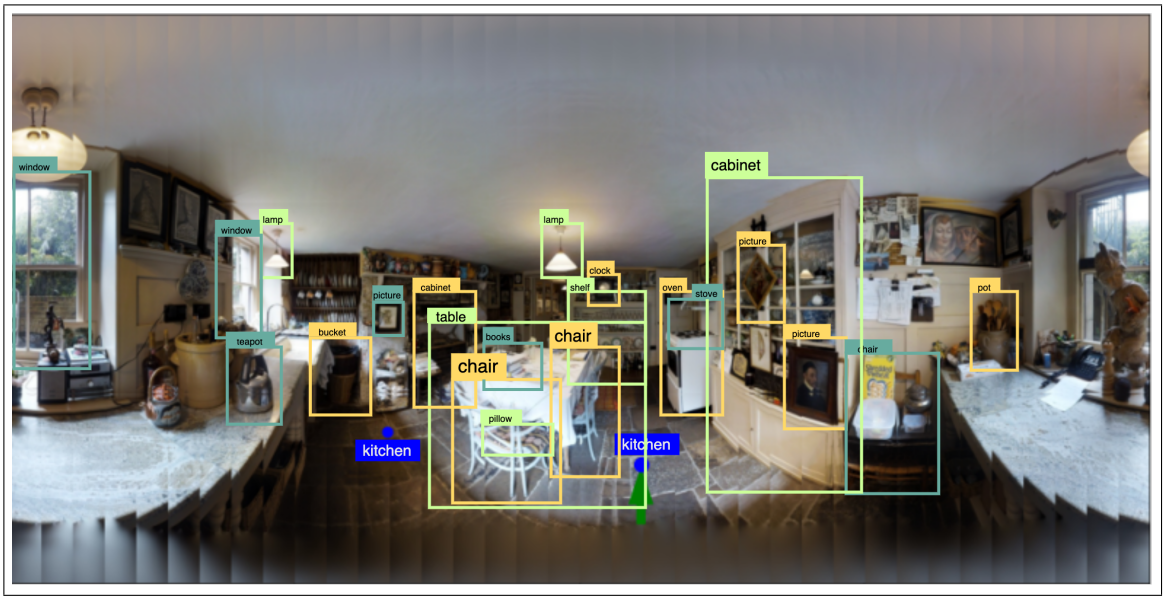


Figure 2.1. Objects and viewpoints visualization through `360-visualization` scripts.

action, and the task is successful if the agent arrives at a location near the target position, recognizing it as the goal.

2.2. Comparison metrics

To compare the performance of presented configurations, we use path length (PL), navigation error (NE) and success rate (SR), as proposed in R2R (Anderson et al., 2018). We also use a new metric called success rate weighted by Path Length (SPL) (Majumdar et al., 2020; Tan et al., 2019; Zhu et al., 2020; Ma, Lu, et al., 2019), that measures the success rate normalized by path length. These metrics are described on Table 2.2.

2.3. New problems

The task proposed by R2R opens new doors to research in mobile robot navigation in real environments. Although the navigation is an important element, it becomes just another task within the wide possibilities that a robot has to interact with household tasks.

Table 2.1. Description of metrics used in this work.

Name	Description
Path Length (PL)	Total distance in meters of the path predicted by the agent
Navigation Error (NE)	Total distance in meters between the goal and the node the agent stopped
Success Rate (SE)	Percentage of times where the agent successfully reach the goal of a path. This goal is represented as an area of three meters from the goal point.
Success Rate weighted by Path Length (SPL)	Success Rate weighted by the normalized Path Length. Measures the efficiency of predicted paths.

Early 2019 brings a new task, called Embodied Visual referring Expression in Real 3D Indoor Environments (REVERIE) (Qi et al., 2020), where an agent must follow an instruction to locate an object in a certain environment, contributing a new dataset that includes annotations and paths to both instructions and object locations (Figure 2.2a).

At the end of 2019, Action Learning From Realistic Environments and Directives (ALFRED) (Shridhar et al., 2020) is proposed, a problem where an agent must manage to perform a sequence of actions at home (after navigating), such as Pick and Place, Clean and Place, among others, proposing a new dataset that through the AI2-THOR 2.0 (Kolve et al., 2017) simulator allows performing more complex actions (Figure 2.2b).

While REVERIE and ALFRED are two tasks that lead an agent to perform actions closer to a future scenario of robots interacting with humans, both rely on visual and language navigation, which remains an open problem.

The authors of R2R have created the BringMeASpoon.org platform, which positions the state-of-the-art models performance on a leaderboard. It is observed that the success rate (SR) does not exceed 73% in unknown environments, but using Beam Search (BS)[1], generating a very high Path Length (PL), which is unrealistic. As we mentioned, under

[1]heuristic that explores a network by selecting the "most promising" node as the target node after knowing all nodes (Ke et al., 2019)

Instruction: Bring me the bottom picture that is next to the top of stairs on level one.

(a) REVERIE task example (Qi et al., 2020)

(b) ALFRED task example (Shridhar et al., 2020)

Figure 2.2. New problems based on the Visual-and-language Navigation Task.

realistic conditions, best performance achieves only 65% of success rate, which tells us that there is still a long way to go in improving the models.

3. RELATED WORK

VLN task has been an active field in Computer Vision, producing surveys and reviews (Wu et al., 2021; Hu et al., 2019) describing several techniques developed over the baseline architecture proposed by R2R.

The main architecture of most models is an encoder-decoder. Having the image of every step and the complete instruction, models have to encode the image along with the instruction, and then feed an LSTM network to sequentially decide the next action.

We group the techniques built on this architecture into categories such as the inclusion of auxiliary tasks (Ma, Lu, et al., 2019; Zhu et al., 2020; Qi et al., 2021; Manterola, 2021; Tan & Bansal, 2019), the improvement of navigation and exploration (Ke et al., 2019; Ma, Wu, AlRegib, Xiong, & Kira, 2019; Ma, Lu, et al., 2019; Wang, Wang, Liang, Xiong, & Shen, 2021; Wang et al., 2020) and curriculum learning with data augmentation (Fried et al., 2018; Majumdar et al., 2020; Tan et al., 2019), which are explained and referenced below.

3.1. Auxiliary tasks

Auxiliary tasks are implemented to correct the models by modifying their loss function. For instance, a progress estimation task has been used by (Ma, Lu, et al., 2019; Zhu et al., 2020; Qi et al., 2021), which estimates the agent's progress in completing an instruction. On each step, this estimation is compared against the real progress, adding a new factor to the loss function. This allows the agent to train his own estimation and then, on an unseen environment, know every time his progress.

Another auxiliary task is the direction prediction task. The agent estimates the angle by which it will turn and it is compared with the exact angle. It has been implemented by Zhu et al. (2020) and Qi et al. (2021). Also, a scene recognition task has been included (Qi et al., 2021; Manterola, 2021), classifying the current step, next step and goal environment

category. Finally, Zhu et al. (2020) includes a trajectory retelling task and a cross-modal matching task, similar to LXMERT (Tan & Bansal, 2019).

3.2. Navigation and exploration

Most of state-of-art algorithms require exploration of the environment which results in a long path, and thus a poor SPL (Success weighted by Path Length).

Therefore, models have proposed to improve the exploration heuristics, such as the technique of self-correction (Ke et al., 2019) algorithm, which improves the beam search method with a back-tracking algorithm using local and global signals from the ecosystem, as well as the Regretful-Agent (Ma, Wu, et al., 2019) algorithm, which uses a rollback oscillation heuristic over the Self-Monitoring model (Ma, Lu, et al., 2019).

Also, the SSM model (Wang et al., 2021) maintains a memory of all seen nodes during the navigation, allowing for a global action space instead of a local one.

Finally, an active exploration and information gathering method has been used to update a node's representation and resolve ambiguity during the navigation (Wang et al., 2020).

3.3. Curriculum learning, pre-training and data augmentation

Commonly, models start training without knowing anything. They are randomly initialized deep learning models. Then, curriculum learning is crucial for models to learn to navigate in complex environments.

Majumdar et al. (2020) proposes to execute curriculum learning with image-text pairs from the web. In effect, the agent acquire knowledge beyond what has been seen in the training of R2R environments, allowing to understand short descriptions of visual images before entering the unknown world of sequenced navigation.

Given the variety of environments and possible navigation instructions, the more data available to feed the neural networks, the better it will adapt to unknown environments.

Accordingly, several works focus on data augmentation. The Speaker-Follower module (Fried et al., 2018), which consists of two models: one that follows instructions (follower) and other that performs data augmentation to feed the training of the follower (speaker), also performing pragmatic reasoning. The speaker module has been used for most state-of-art models in order to generate synthetic data and start with a pre-training phase (Ma, Lu, et al., 2019; Ma, Wu, et al., 2019; Zhu et al., 2020).

Tan et al. (2019) proposes to generate new environments by eliminating objects from the visual scenes, then using back-translation to generate instructions in these new environments.

At last, VLN-BERT+REM (Liu et al., 2021) generates cross-connected house scenes as augmented data via mixuping environment and then pre-train VLN-BERT (Majumdar et al., 2020) with this data.

3.4. Leaderboard

State-of-art leaderboard is summarized in Table 3.1. VLN-BERT+REM (Liu et al., 2021) has the highest success rate (SR), followed by SSM (Wang et al., 2021) and Active Gathering (Wang et al., 2020). These models have high overhead costs due to the time and resources required by their complex architectures. We demonstrate that focusing on text-instructions augmentation, bridging the visual semantic gap, greatly benefits navigation performance without making models even more complex.

Table 3.1. Comparison of the different models solving the Room-to-Room task, in unseen test set using Single Run.

Model	PL ↓	NE ↓	SPL ↑	SR ↑
Speaker-Follower (Fried et al., 2018)	14.82	6.62	0.28	0.35
Tactical Rewind (Ke et al., 2019)	22.08	5.14	0.41	0.54
Self-Monitoring (Ma, Lu, et al., 2019)	17.11	5.99	0.32	0.43
Environmental Dropout (Tan et al., 2019)	11.70	-	0.47	0.51
Regretful-Agent (Ma, Wu, et al., 2019)	13.69	5.69	0.40	0.48
Active Gathering (Wang et al., 2020)	20.6	4.36	0.4	0.58
ORIST (Qi et al., 2021)	10.90	4.72	0.51	0.57
SSM (Wang et al., 2021)	20.7	4.32	0.45	0.62
VLN-BERT + REM (Liu et al., 2021)	13.11	3.87	0.59	**0.65**

4. RESEARCH QUESTIONS AND HYPOTHESES

Current VLN models exhibit limited generalization capabilities, leading to a large drop in performance when they are tested in unseen environments. We believe that lack of a proper visual understanding of the environment is a key factor, leading to our first research question and hypothesis:

RQ1: Is the VLN performance problem due to poor use of visual information on current state-of-art models?

H1: Altering visual features does not greatly variate the performance of navigation models.

Experiments performed on Section 5 indicate that removing visual features, models do not show a large drop in performance. Consequently, we pose a hypothesis and a reserach question to address this problem:

RQ2: How can we contribute to improve the use of visual information in VLN models?

H2: Adding visual information of the environment to training navigation instructions allow models to better understand the world and increase their navigation performance.

5. TESTING THE RELEVANCE OF VISUAL INFORMATION IN VLN

In order to demonstrate models deficits, we experiment in both visual and linguistic modalities using state-of-art models, based on previous analysis (Hu et al., 2019). A summary diagram is shown in Figure 5.1.

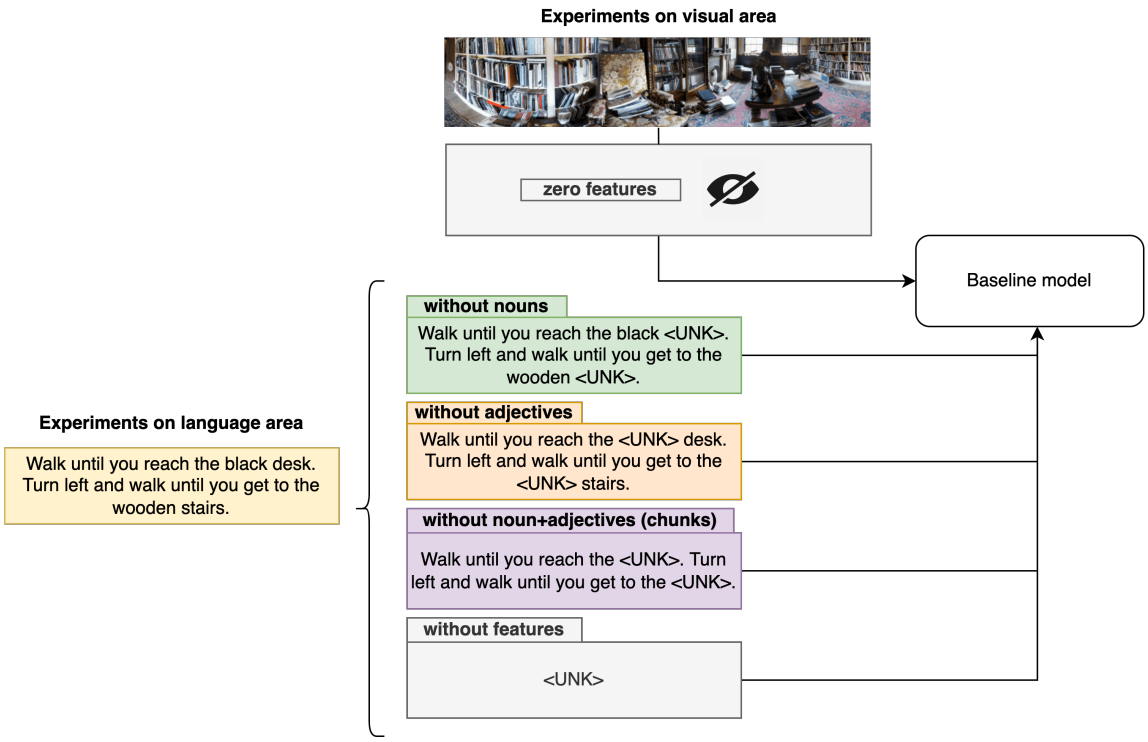


Figure 5.1. Ablation studies setup.

5.1. Experiments in the visual area

To measure the effectiveness of the visual component of current VLN architectures, it is necessary to evaluate the importance of visual information each time the agent decides to execute a specific action.

For experimenting, we use Self-Monitoring (Ma, Lu, et al., 2019) and Regretful-Agent (Ma, Wu, et al., 2019) because of their public codebase[1] [2] and competive performance. Each of these architectures are trained under two different conditions. The first condition is using the base model, where the visual features are obtained from a pre-trained ResNet-152. The second condition is the replacement of visual features with zeros, i.e., the agent is completely blind.

Because both models are built on top of the Speaker-Follower architecture, they also offer an optional pre-training phase that includes training with synthetic data. This synthetic data contains 178,300 sampled routes with associated instructions generated with the Speaker module (Fried et al., 2018). Six experiments evaluated in known (seen) and unknown (unseen) environments were performed. Results are shown in Tables 5.1 and 5.2.

5.2. Experiments in the language area

To measure the effect of the language component in current architectures, we evaluate the importance of different features of the instructions. We use spaCy (Honnibal & Montani, 2017), a NLP model used in the industry, to obtain text features and execute part-of-speech tagging (POS tagging). Each word of each instruction is classified according to the context as adjective, noun, or other. The Self-Monitoring and Regretful-Agent models are trained by extracting words from the instructions: extracting all adjectives, all nouns, all nouns+adjectives, and extracting the whole text (i.e. without linguistic features) from each instruction. We train both models, executing 8 experiments for a total of 100 hours. Results are shown in Table 5.3.

[1] `https://github.com/chihyaoma/selfmonitoring-agent/`
[2] `https://github.com/chihyaoma/Regretful-Agent`
[3] Visual features replaced with zeros

Table 5.1. Visual ablation study on Self-Monitoring and Regretful-Agent, **seen** environments with Single Run.

Model	PL ↓	NE ↓	SPL ↑	SR ↑
Self-Monitoring + `ResNet-152`	13.34	4.02	0.62	0.62
Self-Monitoring + `pre-training` + `ResNet-152`	12.3	3.03	0.63	0.7
Self-Monitoring + `blind`[3]	15.64	7.1	0.23	0.32
Regretful-Agent + `ResNet-152`	12.66	4.18	0.51	0.59
Regretful-Agent + `pre-training` + `ResNet-152`	12.49	3.07	0.63	0.71
Regretful-Agent + `blind`[3]	19.05	7.6	0.14	0.27

Table 5.2. Visual ablation study on Self-Monitoring and Regretful-Agent, **unseen** environments with Single Run.

Model	PL ↓	NE ↓	SPL ↑	SR ↑
Self-Monitoring + `ResNet-152`	15.88	6.47	0.27	0.39
Self-Monitoring + `pre-training` + `ResNet-152`	16.27	5.99	0.30	0.42
Self-Monitoring + `blind`[3]	15.86	6.6	0.24	0.35
Regretful-Agent + `ResNet-152`	16.09	5.99	0.30	0.43
Regretful-Agent + `pre-training` + `ResNet-152`	15.75	5.62	0.35	0.47
Regretful-Agent + `blind`[3]	18.8	6.62	0.19	0.36

Table 5.3. Language ablation study on Self-Monitoring and Regretful-Agent, **unseen** environments with Single Run. Best success rate is in bold.

Model	PL ↓	SR ↑
Training with real data		
Regretful-Agent *baseline*	16.1	**0.43**
Regretful-Agent w/ `nouns`	15.5	0.35
Regretful-Agent w/ `adjectives`	14.8	0.42
Regretful-Agent w/ `nouns+adjectives`	14.9	0.37
Regretful-Agent w/ `all textual features`	18.0	0.25
Training with real data + augmented data		
Regretful-Agent *baseline*	15.8	0.47
Regretful-Agent w/ `nouns`	14.8	0.36
Regretful-Agent w/ `adjectives`	15.5	**0.48**
Regretful-Agent w/ `nouns+adjectives`	13.9	0.39
Regretful-Agent w/ `all textual features`	18.0	0.25

w/ = training with instructions without [`nouns | adjectives | nouns+adjectives | all textual features`]

5.3. Ablation studies results

Our results indicate that, after removing all the visual features, the agents present a completely different behavior when tested in seen or unseen environments. In seen environments the difference in success rate is very large (Table 5.1): while the Self-Monitoring and Regretful-Agent models achieve about 60% of success rate (SR) without pre-training, removing the agent's sight (+ `blind`) greatly reduces its performance (-30%).

In unknown environments the difference is much smaller. We observe that none of the models without pre-training improve more than 7% SR over the blind model. This demonstrates good memorization but lack of generalization, being visual information almost useless when testing on previously unknown scenes.

When experimenting in the linguistic area, we noticed that when we extract the whole text passage, it only reaches a 25% SR. This means that 1 out of 4 random walks actually reaches the goal, noting the biases of the R2R dataset, where agents can navigate correctly to the goal position without any instruction.

If we extract the nouns or nouns+adjectives from the instruction, then the model reduces the SR moderately. This explains that many of the instructions are based on prompts such as "turn right" or "walk straight to the bottom", without necessarily reference the environment.

The removal of adjectives increases the SR, indicating that nouns descriptions are actually interfering with the model performance. We believe the root cause is the high level of abstraction of the instructions generated by humans when collecting the R2R data.

Following the previous observations, in this thesis we propose to create and train with semantically richer instructions, in order to include more detailed description of the visual environment and then force the agent to use all the available information, bridging the visual semantic gap in VLN.

6. SEMANTICALLY RICHER INSTRUCTIONS

To bridge the visual semantic gap in visual and language navigation, we must first learn to follow semantically meaningful instructions that foster the use of visual information. We create simple instructions that use scene objects, referencing them in order to enrich generic and non visual instructions like “go straight”.

As a base model, we use the Speaker module of the Speaker-Follower architecture (Fried et al., 2018). Figure 6.1 shows our complete model. The original Speaker module takes, for each path, the sequence of panoramic views and also the actions sequence (`RIGHT`, `<END>`, `FORWARD`, etc.), and passes them across an encoder module. This encoder provides an encoded context `ctx` that is used for generating each word of the new instruction through an LSTM, which uses also the previous cell and hidden states, as shown in the figure.

R2R dataset has three instructions for each path. During training, the Speaker module builds the loss function for each instruction as the Negative Log Loss (NLL) between the corresponding word of the instruction and the generated word (Fried et al., 2018).

Instructions generated by the Speaker module are now being used for almost all state-of-art models of VLN task on a pre-training phase. However, it has been shown that they do not follow human syntax, they have orientation problems and do not include relevant information, being incorrect in most cases (Zhao et al., 2021).

Using Matterport3DMeta, we propose to add relevant objects to generated instructions feeding the Speaker language model with two loss auxiliary tasks, aiming the vision to be mandatory for the agent to navigate (Figure 6.1):

- **Objects auxiliary task:** The first auxiliary task consists of comparing each word of the generated instruction with the best candidate object according to the path and the next action of the agent. Concatenating the best candidate objects we generate a sentence that we name “objects instruction”.

- **Crafted auxiliary task:** The second auxiliary task consists of comparing the generated instruction with a "crafted instruction". This crafted instruction is generated following a set of rules, according to every action in the path and the objects that surround the agent.

We first explain how we retrieve objects of the environment. Then, the methods we use to generate objects instructions and crafted instructions for a specific path, describing in each one the way we feed the model with the respective auxiliary task.

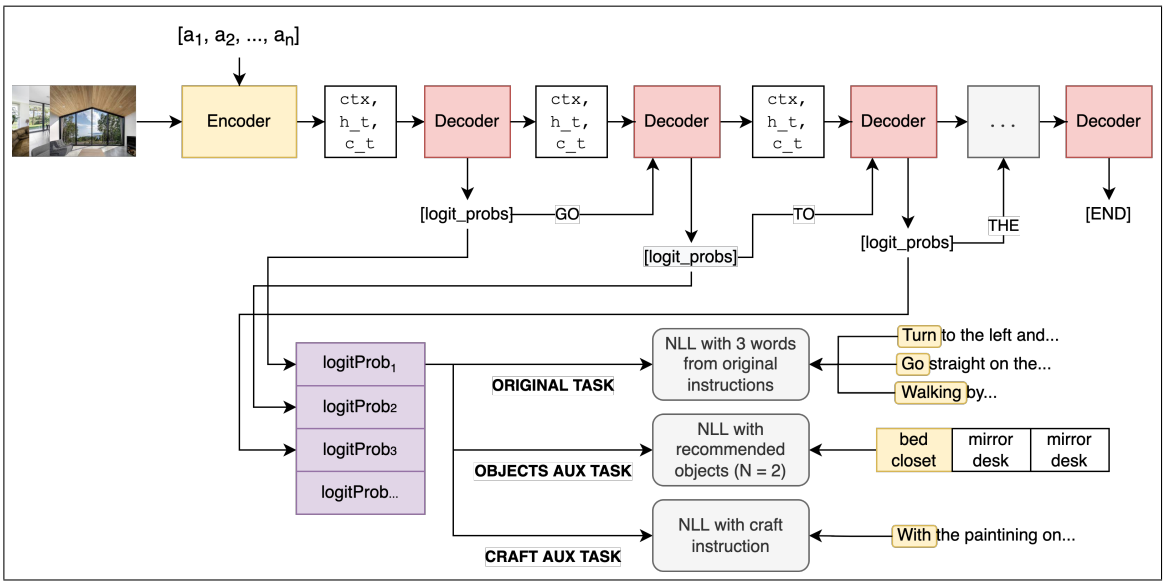


Figure 6.1. Speaker language model architecture with the proposed auxiliary tasks: objects auxiliary task and crafted auxiliary task.

6.1. Objects

Object metadata is available on Matterport3DMeta, but it is raw and difficult to use. That's why we created `360-visualization`[1], a script for fetching and visualizing objects and navigable viewpoints on each node, for each heading and elevation. These objects are the main component for creating objects instructions and crafted instructions. Figure 6.2 shows sampled objects visualized with `360-visualization`.

[1]https://github.com/cacosandon/360-visualization

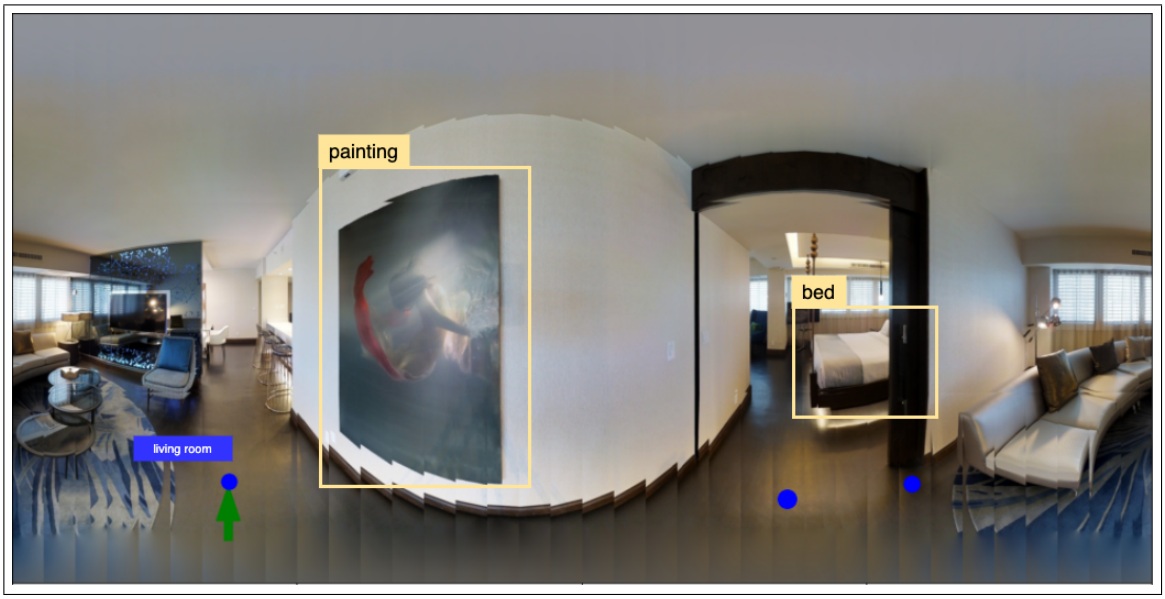


Figure 6.2. Objects and nodes retrieved with `360-visualization` for a specific node on a path.

6.1.1. Metadata parser

We create from the Matterport3DMeta scripts that parse the raw metadata and transform it to clean dataframes. This metadata include all levels, regions, surfaces, vertexes, images, categories, objects, and segments present in a house (Figure 6.3). Most useful dataframes used in this work are described below.

Regions: All regions inside a Scan (house). The resulting dataframe is shown in Figure 6.4. These regions include the center position, the height, and a letter as a label which is mapped to an environment description. This mapping is described in Appendix A.1.

Objects: All objects inside a house. The table in Figure 6.5 shows the dataframe head. These objects include an oriented bounding box, a position, a category index and a region index. The oriented bounding box is used for describing visual boxes, their area and position on the 360 ° image. Fetch of labels, bounding boxes, distance, elevation and heading of objects are described in Appendix A.2.

We also create useful functions to retrieve information directly from these dataframes.

```
metadata.header
```

```
{'name': '-',
 'label': '-',
 'num_images': 1044.0,
 'num_panoramas': 58.0,
 'num_vertices': 201.0,
 'num_surfaces': 22.0,
 'num_segments': 54524.0,
 'num_objects': 563.0,
 'num_regions': 1659.0,
 'num_portals': 22.0,
 'num_levels': 0.0,
 'xlo': 0.0,
 'ylo': -5.65365,
 'zlo': -8.21815,
 'xhi': -1.53947,
 'yhi': 36.8127,
 'zhi': 20.108}
```

Figure 6.3. Scan `Z6MFQCViBuw` raw metadata information.

```
scan = 'Z6MFQCViBuw'
metadata = HouseSegmentationFile.load_mapping(scan)

metadata.regions.head()
```

```
Cached file exists, loading.
```

	region_index	level_index	label	px	py	pz	xlo	ylo	zlo	xhi	yhi	zhi	height
0	0	0	b	2.716100	12.39950	0.004282	-0.405026	10.52090	0.004282	6.70222	17.69280	5.18181	5.17753
1	1	0	n	1.813130	7.21122	-0.000327	-1.107260	5.35311	-0.000327	5.40658	9.59297	5.17126	5.17159
2	2	0	e	0.239733	1.15094	0.006414	-5.653650	-5.24087	0.006414	6.69177	4.47429	5.26821	5.26180
3	3	0	d	9.646000	2.52233	0.003505	7.143930	-2.47150	0.003505	16.73270	9.48628	5.17752	5.17401
4	4	0	n	22.555700	-4.93599	-0.007940	17.184000	-8.15808	-0.007940	36.66280	3.17319	8.72445	8.73239

Figure 6.4. Sampled regions of scan `Z6MFQCViBuw` parsed from raw metadata.

Reachable viewpoints: Standing on a specific viewpoint node inside a house, we can retrieve all reachable viewpoints where the agent can move to. Example information about these viewpoints is shown in Figure 6.6.

Visible objects: Standing on a specific viewpoint node inside a house, we can retrieve all objects present in the scene. Example information about these objects is shown in Figure 6.7.

```
metadata.objects.head()
```

	object_index	region_index	category_index	px	py	pz	a0x	a0y	a0z	a1x	a1y	a1z	r0	r1	r2
0	0	0	2	3.08146	13.8466	5.02184	1	0	0	0	1	0	3.30238	3.13547	0.236681
1	1	0	176	-0.0262976	12.2344	0.294519	0	1	0	1	-0	0	0.38656	0.308878	0.276866
2	2	0	766	0.0668365	13.8391	1.95553	0	0	1	0	1	-0	1.94078	1.47311	0.518241
3	3	0	6	4.22211	11.6596	0.500373	0	0	1	-0.707107	0.707107	0	0.477757	0.337069	0.311606
4	4	0	7	6.487	13.7987	1.78647	0	0	1	2.22045e-16	1	-0	1.76339	1.36331	0.447083

Figure 6.5. Sampled objects of scan `Z6MFQCViBuw` parsed from raw metadata.

```
scan = 'Z6MFQCViBuw'
viewpoint_name = '8eda0abb2e714080a7a5d15c9d606c25'
connectivity_path = f'/home/{IALAB_USER}/repos/360-visualization/connectivity/{scan}_connectivity.json'

metadata = HouseSegmentationFile.load_mapping(scan)
reachable_viewpoints = metadata.angle_relative_reachable_viewpoints(viewpoint, connectivity_path)

reachable_viewpoints
```

```
Cached file exists, loading.
```

	distance	elevation	heading	name
0	3.677166	-0.364760	0.461250	c27b2a83f05649e08baae0328c5583db
1	1.587426	-0.717909	-1.573467	bc9bd723cae547c290bb24692fd1b8ce
2	2.289498	-0.549206	2.507270	bc4ec1f735f3446aa4b56165d9508b45
3	1.145663	-0.882938	1.442176	2008e72476f84104858e908beeac0193

Figure 6.6. Reachable viewpoints information of a sampled viewpoint parsed from raw metadata.

6.2. Objects instructions

For each node of a path sequence, we fetch all objects with `360-visualization` and filter them by distance, area, uniqueness and usability (excluding outlier objects, for example, “floor” that has a large area).

We assign the best `N` objects to each word of the instructions of that path, matching the word index with the closer node index. For instance, in Figure 6.8 we recommend the model to use “painting” (`N = 1`) for the first words of the generated instruction.

```
scan = 'Z6MFQCViBuw'
viewpoint_name = '8eda0abb2e714080a7a5d15c9d606c25'
connectivity_path = f'/home/{IALAB_USER}/repos/360-visualization/connectivity/{scan}_connectivity.json'

metadata = HouseSegmentationFile.load_mapping(scan)
reachable_viewpoints = metadata.angle_relative_viewpoint_objects(viewpoint)

reachable_viewpoints[reachable_viewpoints.columns[::-1]].head()
```

```
Cached file exists, loading.
```

	elevation	heading	distance	category_mapping_name	r2	r1	r0	a1z	a1y	a1x	a0z	a0y	a0x	pz	py	px	categor
0	0.784495	1.358218	3.575780	ceiling	0.267166	4.31009	5.63545	0	-0	1	0	1	0	3.56933	0.75442	3.49529	
1	0.177012	2.516195	5.982186	wall	0.16429	2.43313	4.41125	1	0	0	0	2.22045e-16	-1	1.07012	-4.84994	3.50209	
2	0.147371	0.506560	7.221335	wall	0.189802	2.44672	4.38637	1	0	0	0	2.22045e-16	-1	1.07199	6.31447	3.50359	
3	0.133617	1.476722	7.833126	wall	0.135927	2.44083	5.70541	1	0	0	0	1	0	1.05291	0.73581	7.79849	
4	0.733564	-0.833687	1.184215	wall	0.324631	2.45136	5.69522	1	0	0	0	1	0	1.06740	0.79597	-0.87681	

Figure 6.7. Visible objects information of a sampled viewpoint parsed from raw metadata.

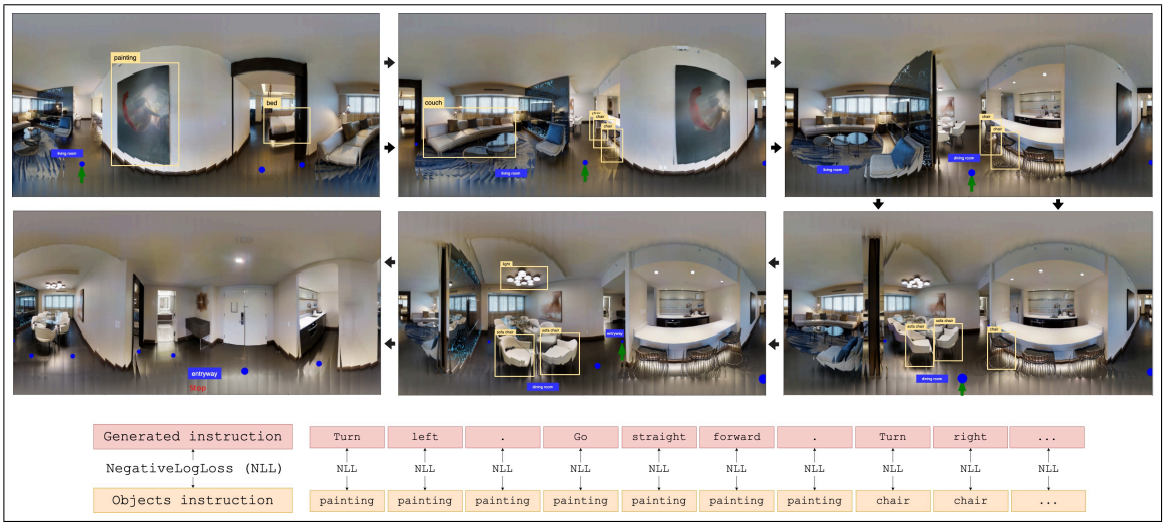


Figure 6.8. Objects instruction generated for the path and the resulting negative log loss applied to the generated instruction on training.

As shown in the figure, the objects auxiliary task consists on adding Negative Log Loss between the generated word and the N recommended objects to the final loss. The sum of these losses are weighted by λ, a modifiable parameter. The final loss when training the Speaker with the objects auxiliary task is shown in Equation 6.1.

$$wordLoss = \sum_{i=1}^{3} NLL\left(log(logit), w_{original_i}\right) + \lambda \sum_{i=0}^{N} NLL\left(log(logit), w_{object_i}\right) \quad (6.1)$$

6.3. Crafted instructions

We create crafted instructions in order to feed the Speaker module with the crafted auxiliary task and also to use them directly as the set of instructions for training navigation agents. For a specific path, we generate an atomic instruction for each node on the sequence. Having the current 360 ° visual image we can create a short instruction based on the orientation of the next node (Figure 6.9) and objects present in the path (Figure 6.2), following a set of rules. These rules and the way we generate crafted instructions are explained in more detail in Appendix B.1. For instance, in Figure 6.10 we start with a big painting at the right of the next node, generating the first atomic instruction: "Turn left, walk straight down the left of the painting". Then, we concatenate all this atomic instructions, generating a new crafted instruction for the selected path.

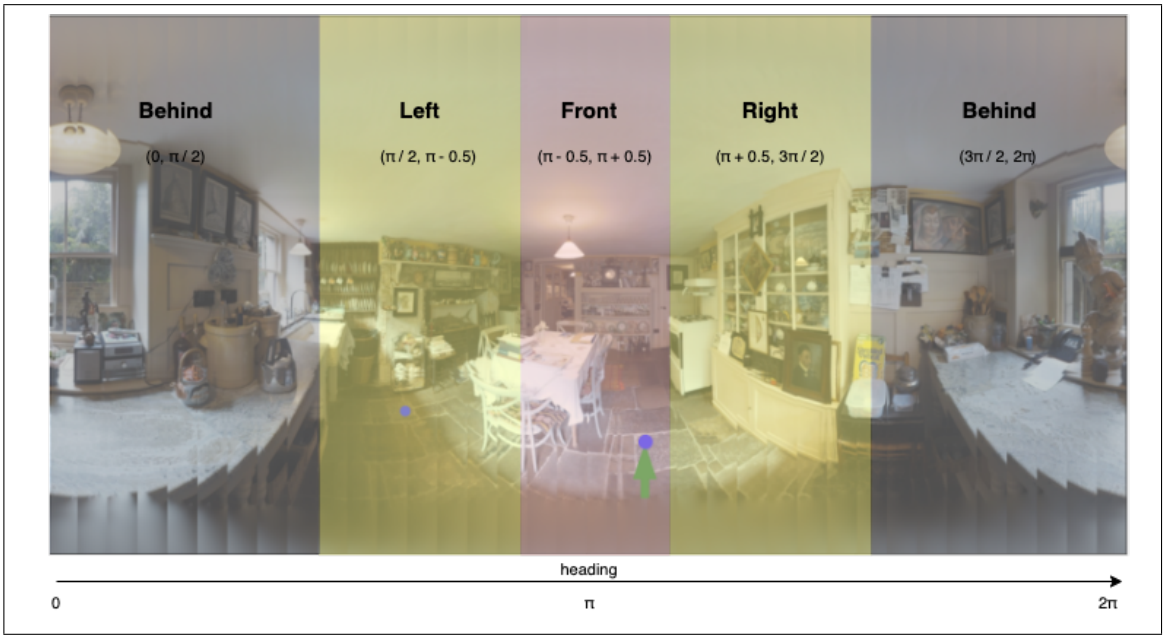


Figure 6.9. Agent orientation mapping on a 360 ° image.

HUMAN 1: Walk past the large picture and chair. Walk past the dining room table turn right into the hallway and stop.

HUMAN 2: Go straight and pass the bar with the chair/stools then pass the clear glass table with the white chairs and turn right. Wait in that hallway.

HUMAN 3: With the painting of the mermaid towards your right, head straight. After passing the counter towards your right, turn right and wait.

CRAFTED: Take a left, walk straight down the left side of the painting. Walk forward. Exit the living room to the dining room. Go out of dining room into the entryway walking with the sofa chair on your left. Stop.

Panoramic views sequence at top and human instructions + crafted instruction at bottom. Images are sequenced through the arrows. Presented objects and scenes names are sampled using `360-visualization`.

Figure 6.10. Crafted instruction example.

As an auxiliary task, we add a Negative Log Loss between the generated instruction and the crafted instruction, word by word. An illustration is shown in Figure 6.11. The sum of these losses are weighted by β, another modifiable parameter. The final loss for each generated word is shown in Equation 6.2.

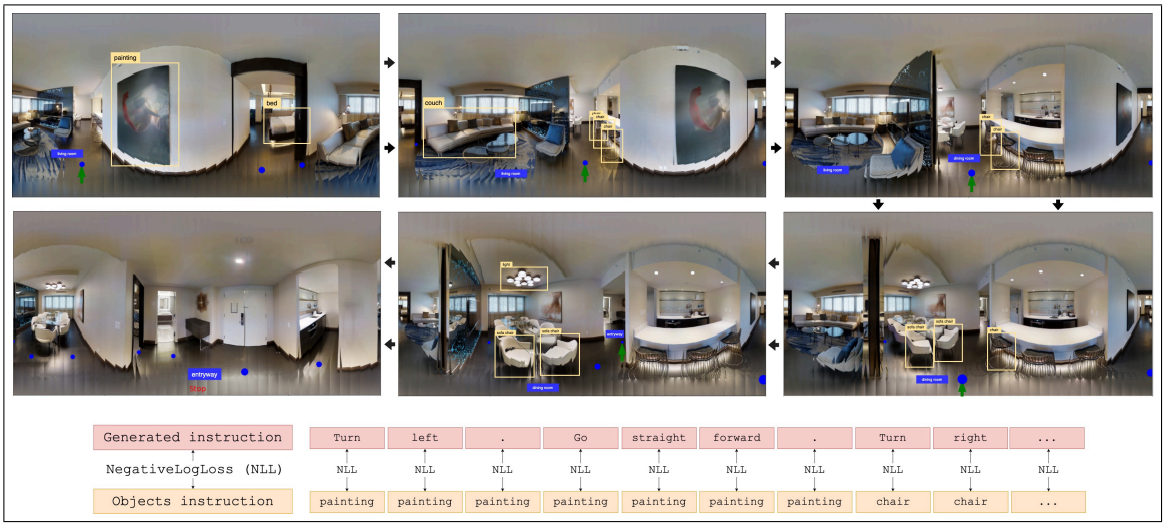


Figure 6.11. Crafted instruction generated for the path and the resulting negative log loss applied to generated instruction on training.

$$wordLoss = \sum_{i=1}^{3} NLL\left(log(logit), w_{original_i}\right) + \beta \cdot NLL\left(log(logit), w_{crafted}\right) \quad (6.2)$$

7. RESULTS AND DISCUSSION

The synthetic data generated with the original Speaker is constructed based on 178,300 sampled paths (Fried et al., 2018). We use the same paths to generate new instructions in several ways:

- **Speaker trained with Objects Auxiliary Task:** Based on Equation 6.1, we feed the Speaker model training with objects auxiliary task, with different values of λ. Once the Speaker is trained, we generate an instruction for each sampled path.
- **Speaker trained with Crafted Auxiliary Task:** Based on Equation 6.2, we feed the Speaker model training with objects auxiliary task, with different values of β. Once the Speaker is trained, we generate an instruction for each sampled path.
- **Crafted instructions directly:** We use our crafted instruction generator to build an instruction for every sampled path, without using the Speaker model.

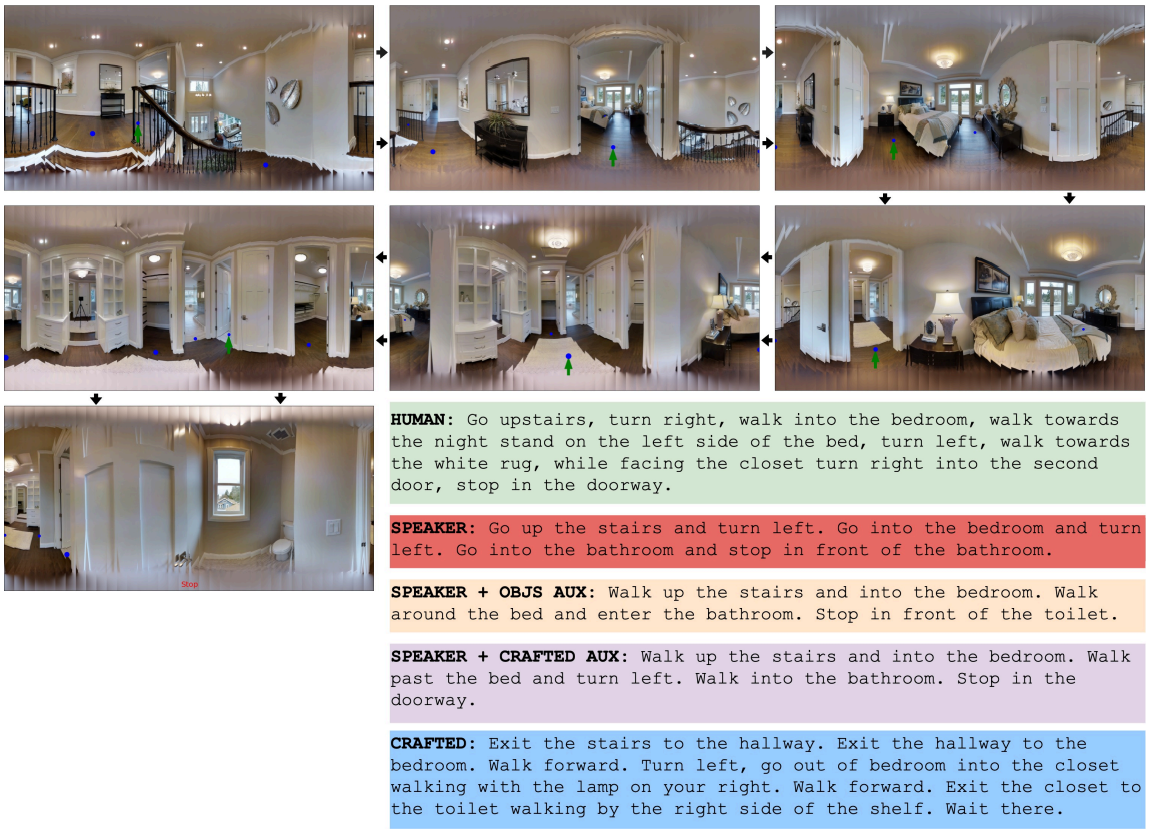
It is important to mention that we train with separated auxiliary tasks, because it is redundant information. Crafted instructions used in the crafted auxiliary task are built with the same objects as the ones we pass directly on the objects auxiliary task.

We demonstrate the quality of the different sets of generated instructions through three modalities: qualitative analysis, evaluation of the Regretful Agent pre-trained with these instructions and assessment of human wayfinding.

7.1. Qualitative analysis of generated instructions

We generate semantically richer instructions, having several advantages. First, as shown in Figure 7.1, it corrects the Speaker module in the orientation, since we help the model by indicating which objects to reference in the instruction. For instance, in the

figure's sequence it is clear that the agent must turn right, while the Speaker generates an instruction that wrongly says the opposite.



Panoramic views sequence above and human instruction + speaker follower instructions + speaker follower with objects auxiliary task + speaker follower with crafted instructions auxiliary task + crafted instruction below. Images are sequenced through the arrows.
Our generated instructions correct the orientation of the Speaker base. For instance, the Speaker base generates an instruction that says "turn left", which is wrong. Instruction from Speaker with objects auxiliary task includes multiple relevant and unused objects, while the instruction from Speaker with crafted auxiliary task imitates the rules used for generating crafted instructions, being simple and with a low level of abstraction.

Figure 7.1. Comparison of human instruction with instructions generated using the Speaker base model, instructions generated using the Speaker with auxiliary tasks and our crafted instructions.

Second, compared to human instructions we realize that a longer instruction is not entirely necessary, since with a short one we can reach the goal. Humans describe a route in great detail, which makes it even more complicated for the agents. In the VLN task where there is so much information, the agent should first learn basic and feasible

navigation with simple instructions, and then incorporate more and more information. A good starting point is to begin training with our model generated instructions.

Finally, our generated instructions reference relevant objects that are in the environment and are not referenced by human instructions, as it does with the word “toilet” at the end of the instruction. The model learned to use objects even though it has never seen them before, being able to detect objects in unknown environments.

7.2. Evaluation of the Regretful Agent pre-trained with generated instructions

In order to compare the quality of the different sets of generated instructions, we train the Regretful Agent (Ma, Wu, et al., 2019), a state-of-art navigation model, using the generated synthetic data as pre-training, ranking its navigation performance in unseen environments.

The complete training consists on, as a first step, pre-train the agent with synthetic instructions, which are generated using the Speaker base, the Speaker with the proposed auxiliary tasks or the crafted instructions generator. After pre-training, we perform a training phase using the original R2R instructions. Results for seen and unseen environments are shown in Tables 7.1 and 7.2.

Table 7.1. Regretful-Agent pre-training with instructions generated from different sources, evaluated on **seen** environments with Single Run and original human instructions. Best success rate is in bold.

Regretful-Agent +	PL ↓	NE ↓	SPL ↑	SR ↑
`Without pre-training`	12.66	4.18	0.51	0.59
`PWIF Speaker base`	12.49	3.07	0.63	0.71
`PWIF Speaker + Objects AT` $\lambda = 0.3, N = 2$	12.97	3.10	0.62	0.71
`PWIF Speaker + Objects AT` $\lambda = 0.5, N = 1$	11.65	3.38	0.61	0.67
`PWIF Speaker + Objects AT` $\lambda = 0.5, N = 2$	12.09	2.93	0.65	**0.72**
`PWIF Speaker + Objects AT` $\lambda = 0.5, N = 3$	12.80	3.48	0.58	0.67
`PWIF Speaker + Objects AT` $\lambda = 0.6, N = 2$	12.07	3.09	0.63	0.70
`PWIF Speaker + Crafted AT` $\beta = 0.1$	12.41	3.16	0.62	0.70
`PWIF Speaker + Crafted AT` $\beta = 0.2$	11.83	3.29	0.62	0.68
`PWIF Speaker + Crafted AT` $\beta = 0.3$	12.24	2.86	0.63	**0.72**
`PWIF Speaker + Crafted AT` $\beta = 0.4$	12.16	3.08	0.62	0.70
`PWIF Crafted directly`	12.50	3.32	0.59	0.67

`PWIF` = *Pre-training with instructions from*, `AT` = *Auxiliary Task*.

Table 7.2. Regretful-Agent pre-training with instructions generated from different sources, evaluated on **unseen** environments with Single Run and original human instructions. Best success rate is in bold.

Regretful-Agent +	PL ↓	NE ↓	SPL ↑	SR ↑
`Without pre-training`	16.09	5.99	0.30	0.43
`PWIF Speaker base`	15.75	5.62	0.35	0.47
`PWIF Speaker + Objects AT` $\lambda = 0.3, N = 2$	15.27	5.39	0.36	0.49
`PWIF Speaker + Objects AT` $\lambda = 0.5, N = 1$	14.66	5.80	0.35	0.46
`PWIF Speaker + Objects AT` $\lambda = 0.5, N = 2$	14.61	5.29	0.39	**0.51**
`PWIF Speaker + Objects AT` $\lambda = 0.5, N = 3$	15.24	5.77	0.34	0.47
`PWIF Speaker + Objects AT` $\lambda = 0.6, N = 2$	15.82	5.46	0.34	0.48
`PWIF Speaker + Crafted AT` $\beta = 0.1$	14.90	5.75	0.35	0.47
`PWIF Speaker + Crafted AT` $\beta = 0.2$	14.37	5.58	0.38	0.48
`PWIF Speaker + Crafted AT` $\beta = 0.3$	15.42	5.52	0.37	0.50
`PWIF Speaker + Crafted AT` $\beta = 0.4$	15.44	5.43	0.36	0.47
`PWIF Crafted directly`	15.97	6.03	0.33	0.46

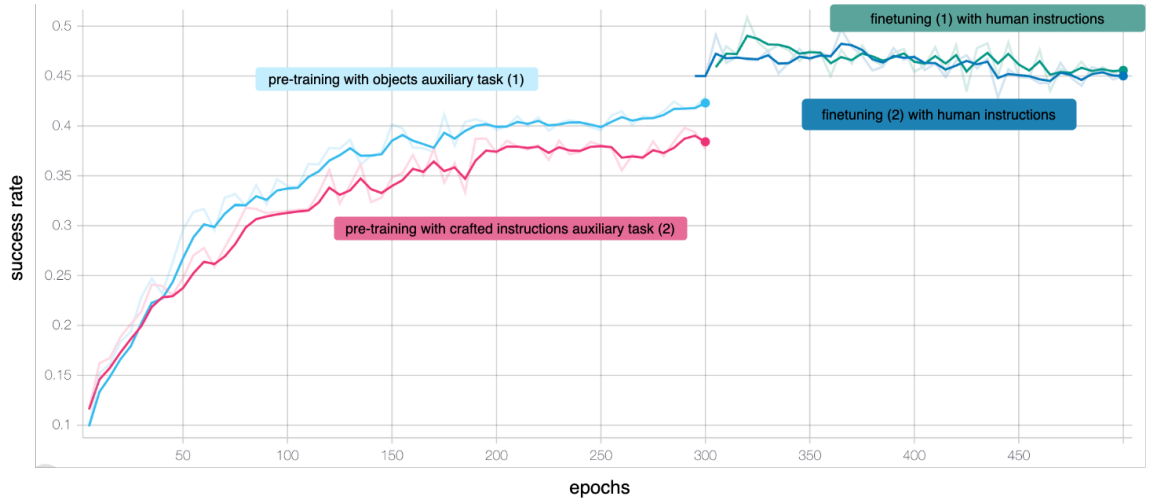
`PWIF` = *Pre-training with instructions from*, `AT` = *Auxiliary Task*.

As shown in the result tables and according to Equations 6.1 and 6.2, we train the Speaker base model with different values for λ and β, varying the weight of the auxiliary task on the loss function for each training configuration. For the objects auxiliary task, we also test with different values of N, the number of objects suggested for each word. For each configuration, we generate a new set of synthetic instructions, which is used in the pre-training phase of the Regretful Agent. Best results are achieved by:

- Pre-training with generated instructions from the Speaker module including objects auxiliary task weighted by $\lambda = 0.5$ and with $N = 2$ (objects per word).

- Pre-training with generated instructions from the Speaker module including crafted auxiliary task weighted by $\beta = 0.3$.

We increase the success rate (SR) up to 72% on seen environments and up to 51% on unseen environments by pre-training with our generated instructions. On the other hand, we increase the success rate weighted by the path length (SPL) up to 65% on seen environments and up to 39% on unseen environments. Figure 7.2 shows the values of the SR by testing on unseen environments during the pre-training (leftmost curves) and training (rightmost curves) using the best two sets of synthetic data generated with the best two configurations described above.



Leftmost curves (light blue & pink) represent pre-training with the generated instructions (synthetic data) and rightmost curves (blue & green) represent training with original R2R data (human instructions).

Figure 7.2. Success rate of Regretful-Agent pre-trained with instructions generated with Speaker trained with the two best configurations of λ and β on unseen environments.

Table 7.3 summaries the comparative performance of the Regretful Agent trained with different sets of instructions. The agent trained with the instructions generated from the

Speaker module with the best objects auxiliary task configuration ($\lambda = 0.5$, $N = 2$) reaches 51% on SR, and 39% on SPL when testing on unseen environments. This is an increment of 8% and 9%, respectively, doubling the increment of the original synthetic data from the Speaker base (4% on SR and 5% on SPL).

Table 7.3. Summary table of the performance of the Regretful-Agent pre-trained with instructions generated from different sources, evaluated on **unseen** environments with Single Run and original human instructions.

Regretful-Agent +	SPL ↑	SR ↑
`Without pre-training`	0.30	0.43
`PWIF Speaker base`	0.35 (+5%)	0.47 (+4%)
`PWIF Speaker + Objects AT` $\lambda = 0.5, N = 2$	0.39 (+9%)	0.51 (+8%)
`PWIF Speaker + Crafted AT` $\beta = 0.3$	0.37 (+7%)	0.50 (+7%)
`PWIF Crafted directly`	0.33 (+3%)	0.46 (+3%)

`PWIF` = *Pre-training with instructions from*, `AT` = *Auxiliary Task*.

Emphasising that the Speaker follower introduces a pre-training phase with 178,300 instructions, increasing the original number of 4,675 human instructions by 173,625, results demonstrate that the SR increment using original synthetic data from Speaker base is not mainly based on the quality of the Speaker's instructions, but rather on the quantity, which is the main contributor for achieving a better performance.

As we mention, using the same 178,300 sampled paths, we create totally new crafted instructions without needing human instructions or the Speaker module, and use them also when pre-training the Regretful Agent. On Tables 7.1, 7.2 and 7.3 it is shown as "`PWIF Crafted Directly`". We almost reach the same success rate as pre-training with original synthetic instructions and we exceed base training by 4%. We then demonstrate that

the Speaker module as a language model does not contribute more than instructions generated based on rules, unless we add the proposed auxiliary tasks, where the performance difference is remarkable.

Our artificially generated navigational instructions provide a significant boost in performance when training the Regretful Agent. They also reduce the visual semantic gap between seen and unseen environments, demonstrating the contribution to the use of visual information when navigating in unknown environments.

7.3. Assesment of human wayfinding

We build a page where humans can navigate in different environments in the same way that a robot would (Figure 7.3), using instructions from multiple sources. For a random path of an unseen environment, we randomly select one instruction of the following sources: `Human`, `Crafted (ours)`, `Speaker` (Fried et al., 2018), `Speaker + Objects Auxiliary Task (ours)` and `Speaker + Crafted Auxiliary Task (ours)` and ask a human to follow it through executing actions.

This web page is written in the *Python* language, specifically in the *Flask* framework. The 360 ° panoramic views with all the necessary information are obtained through `360-visualization`, our script developed for this work. Code of the platform is available on Github [1].

The platform allows the execution of any action that an agent could execute in an R2R navigation (Figure 7.4): rotate, move and stop. We describe the main features of the platform below.

[1] https://github.com/cacosandon/instructions-follower

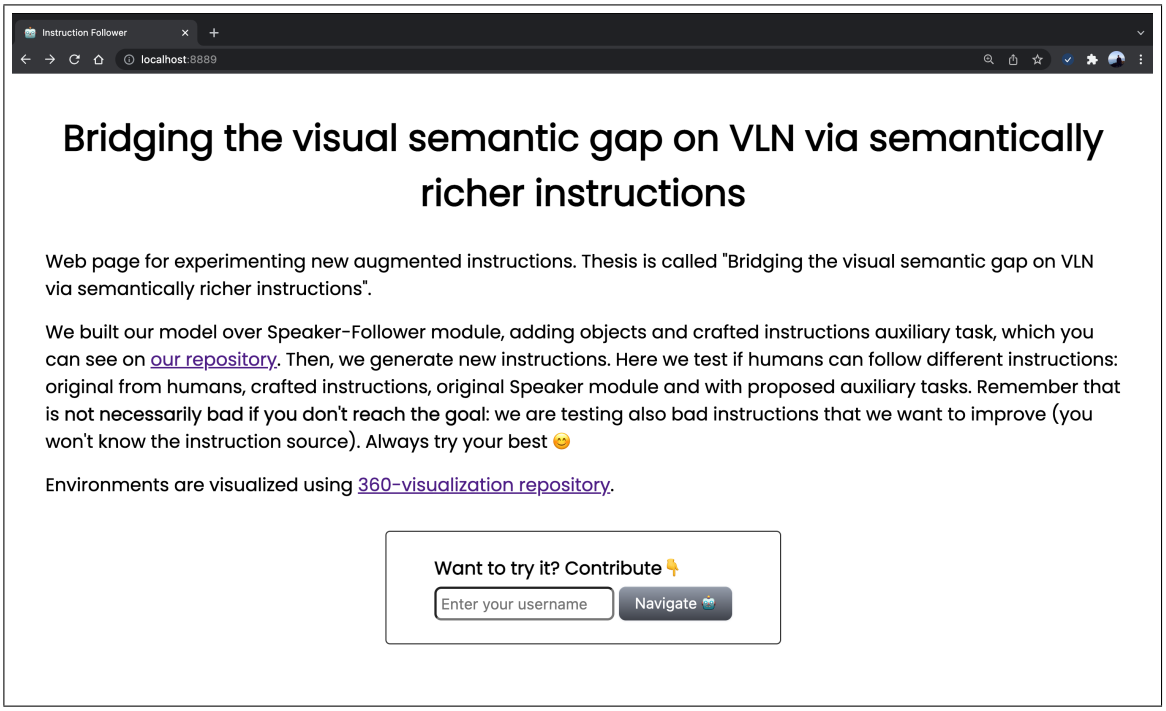


Figure 7.3. Landing of the instruction follower platform.

7.3.1. Panoramic view

We use panoramic (360 °) images, simulating the same conditions of navigation agents. As shown in Figure 7.4, angles are specified on the horizontal axis. -90 ° means left, 90 ° means right and (-)180 ° means behind.

7.3.2. Rotate

Humans using the platform can rotate left, right or turn around. On Figure 7.5 we see how the bed moved from being on the left of the image (-90 °) to the center after turning left.

7.3.3. Move to navigable node

In order to navigate to the goal, platform allows humans to move through nodes. In Figure 7.6 we show the sequence when human select to move to node “2”.

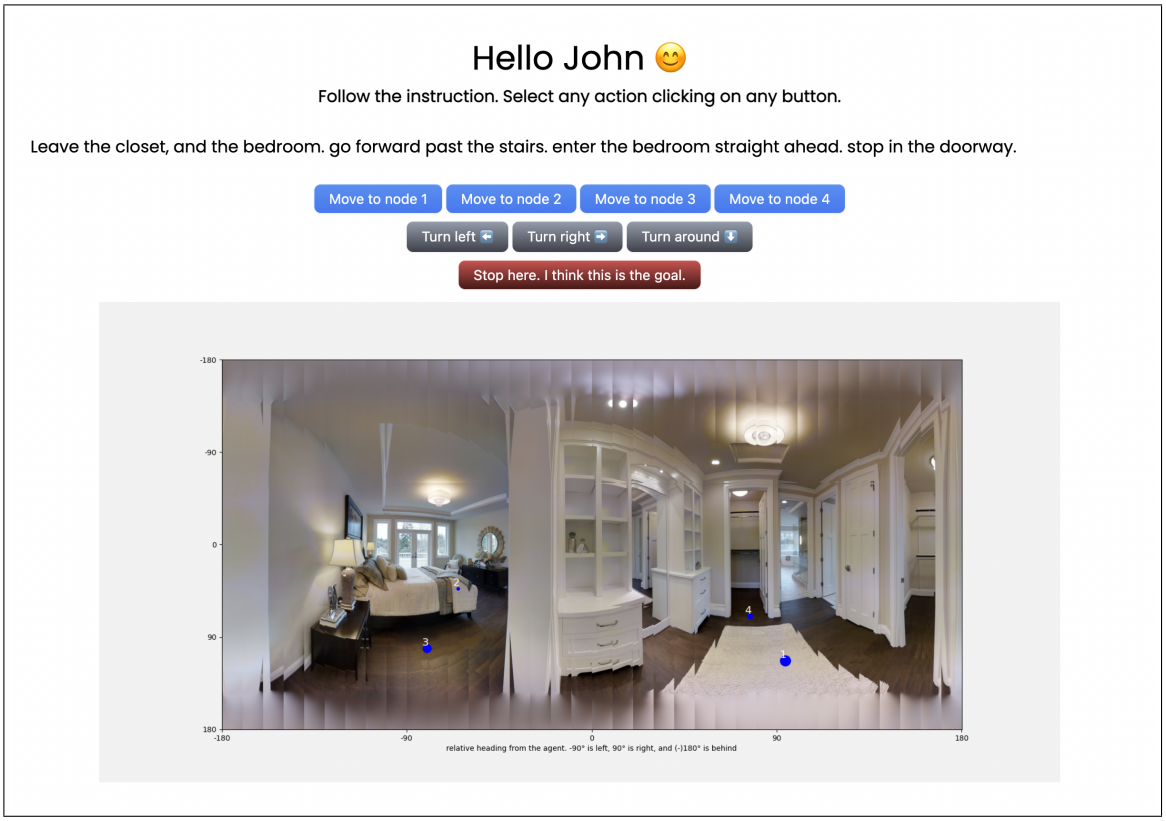


Figure 7.4. Available actions on the instruction follower platform.

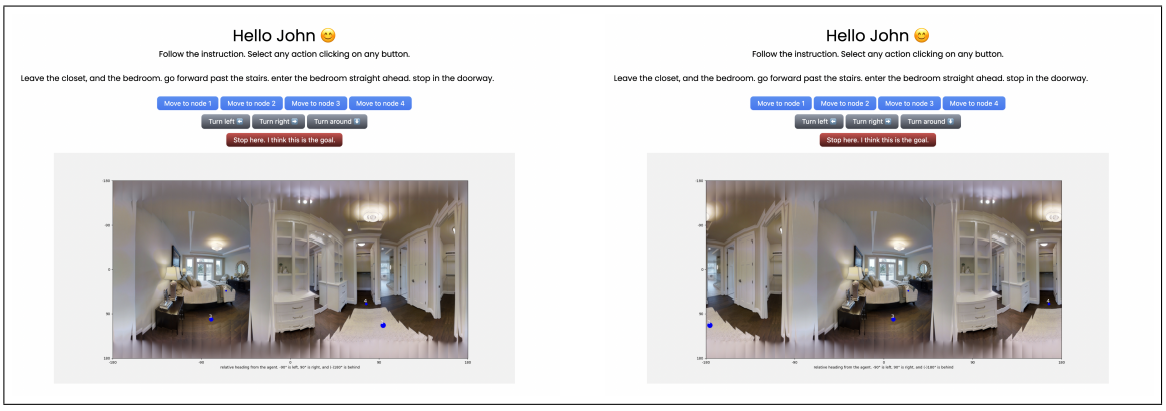


Figure 7.5. Sequence after rotating left on instruction follower platform

7.3.4. Stop and retrieve metrics

After navigating, followers can select the stop option. This triggers the computation of metrics and the success flag. For instance, in Figure 7.7 the human stopped at the correct

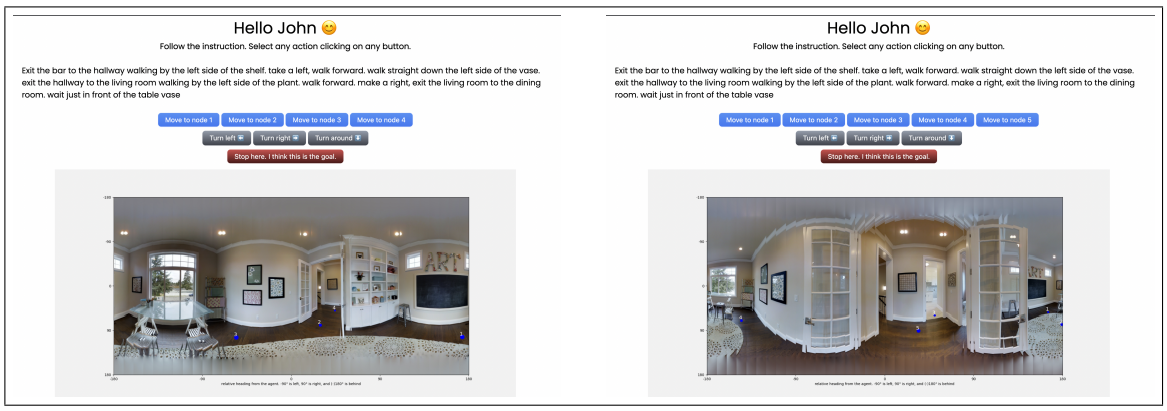


Figure 7.6. Sequence after moving to node "2" on instruction follower platform

node, with a trajectory length of 15.67 meters and a navigation error of 0 meters (stopped in the exact goal node).

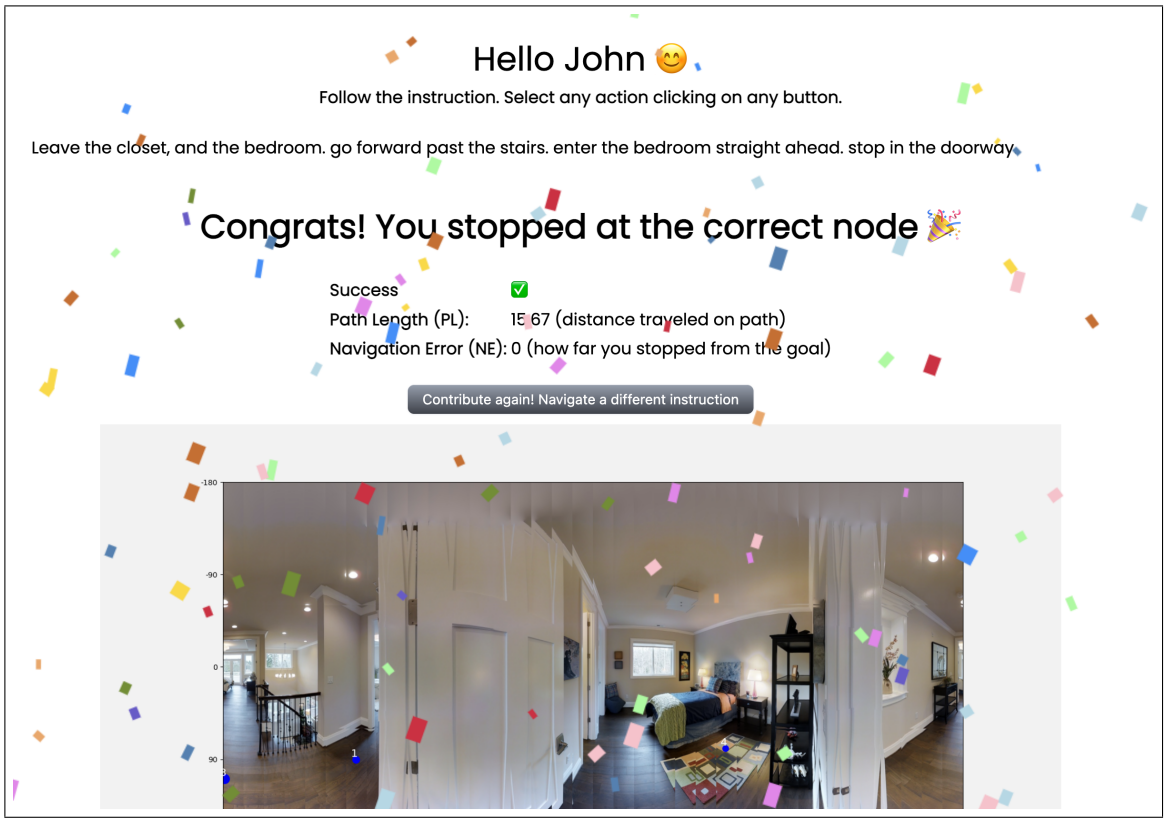


Figure 7.7. Successful navigation on instruction follower platform.

The collected data is saved for post processing. In Figure 7.8 we show a sample of the JSON output. We save the instruction, the source model, the path ID with the scan (house), the owner (name of the human that followed the instruction), the success flag, the path length (TL), the navigation error (NE), the success rate weighted by path length (SPL) and the path containing actions, the accumulated distance, the next node name and the heading.

Figure 7.8. JSON output sample after finishing route on instruction follower platform.

7.3.5. Collected data

We collected a total of $N_{FP} = 200$ followed paths, with a total of 9 people who participated, who we call "followers". They are asked to enter the platform and follow the instructions, without knowing the source of them. Paths are obtained from the validation unseen dataset of R2R, in order to retrieve instructions generated with models trained with other environments.

Because there are limited environments, we calculate the average metrics with the first 15 observations of each follower. After 15 followed paths we realized that the environments became familiar for the followers, biasing the decisions on each node. For example, after several samples, followers start to reach rooms that were specified in the instruction, but not seen in the image. Average metrics are shown in Table 7.4.

Table 7.4. Average metrics of the first 15 observations for each follower when navigating in unseen environments.

Instructions sources +	PL ↓	NE ↓	SPL ↑	SR ↑
`Human`	14.90	4.26	0.59	0.70
`Speaker base`	14.10	6.00	0.30	0.30
`Crafted`	11.66	2.47	0.70	**0.78**
`Speaker + Objects AT` $\lambda = 0.5, N = 2$	14.63	4.15	0.36	0.47
`Speaker + Crafted instructions AT` $\beta = 0.3$	9.39	4.07	0.51	0.61

`AT` means *Auxiliary Task*.

The highest success rate is reached using our crafted instructions, even performing better than using human instructions. By construction, our crafted instructions are meant to be followed by humans, which is reflected in the results. The same happens when we add crafted auxiliary task to the Speaker module, having a considerable success rate difference (+30%) compared to the instructions from the Speaker base.

Although the instructions generated by all our models lead to better human navigation performance than the instructions from the base Speaker module, the Speaker with objects auxiliary task is the worst of our instruction generation models for orientating humans. However, as we show on the previous section, pre-training the Regretful Agent with instructions from this model allows the agent to achieve the best success rate. This indicates that these synthetic instructions are optimized for beginner navigation agents, such as the Regretful Agent, since they are DL models initialized with random values. In other words, instructions from language models (Speaker) are generated for guiding DL navigation agents (Regretful Agent) without necessarily being useful for human navigation. On the other hand, human instructions are optimized for expert navigators, because of their high level of abstraction and complexity, being hardly complex for a DL model to follow.

8. CONCLUSIONS

In this thesis, we use different methodologies to improve scene understanding, in order to achieve a better performance in navigation with human interaction. State-of-art models focus mainly on model architecture, leaving aside the base of the task: the dataset. As we present, navigation agents do not use available visual information as much as they should for making a decision. By removing visual features on training, when testing in seen environments the success rate decreases around 30%, while testing in unseen environments it generates a slight drop of only 7%, evidencing that DL based navigation agents memorize low level correlations between textual and visual data.

In addition, these same instructions are too complex and high level, confusing agents that start as beginners on navigation. To bridge this visual semantic gap, we create new semantically richer instructions. For this purpose, we use scene objects and crafted instructions to feed a set of auxiliary tasks: objects auxiliary task and crafted auxiliary task. As we present, the resulting model generates new instructions that help to correct the errors existing in the original instructions. Using them in the pre-training process of a state-of-art model, we increase the success rate from 43% to 51% (+8%) in unseen environments, which doubles the increase of pre-training with instructions from the original Speaker (+4%).

We also test generated and crafted instructions through a web platform, where humans are asked to reach a target point by following an instruction through a simple interface. Results indicate that, by construction, our crafted instructions perform better than any language-model generated instruction when tested in humans, while instructions generated with the Speaker and our variants perform better in deep-learning-based navigation agents. This is, better performance is achieved when humans follow instructions generated by humans (or with human logic rules, i.e. crafted instructions) and when deep-learning navigation models follow artificially instructions generated with language models.

Through this thesis and its results, we demonstrate that the creation of semantically richer instructions that include explicit and detailed visual information allows navigation agents to better learn to navigate when using them for training, bridging the visual semantic gap in the visual and language navigation task.

9. FUTURE WORK

In order to follow this same line to improve robot navigation, we propose different branches for further research:

Add Object detection: In this work we construct our auxiliary task based on available metadata of different environments (Matterport3DMeta). If we want to expand to new environments where this metadata is not available, we must detect objects on our own. Indoor object detection is an unresolved task, which can be improved directly using the same scene objects that we retrieve from raw data.

3-phase Curriculum Learning: We pre-train our model with our semantically richer instructions, and then finetune with the original instructions, which are complex and high level. Starting with an easier task will allow the agent to use environment information progressively. Standing in a random node, we have the 360° image, different possible navigation nodes and an atomic instruction. The agent has to decide which node to move to. The agent will learn simpler and shorter instructions that refer to the environment, the basics for starting to execute this tasks on sequence.

REFERENCES

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., ... van den Hengel, A. (2018). Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., & Parikh, D. (2015). VQA: Visual Question Answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).*

Arpit, D., Jastrzundefinedbski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., ... Lacoste-Julien, S. (2017). A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning*.

Bekey, G. A. (2005). *Autonomous robots: From biological inspiration to implementation and control*. MIT Press.

Belkin, M., Hsu, D. J., & Mitra, P. (2018). Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. In *Advances in neural information processing systems.*

Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., ... Zhang, Y. (2017). Matterport3d: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*.

Fried, D., Hu, R., Cirik, V., Rohrbach, A., Andreas, J., Morency, L.-P., ... Darrell, T. (2018). Speaker-Follower Models for Vision-and-Language Navigation. In *Neural Information Processing Systems (NeurIPS).*

García, S., Strüber, D., Brugali, D., Berger, T., & Pelliccione, P. (2020). Robotics software

engineering: A perspective from the service robotics domain. In *Proceedings of the 28th acm joint meeting on european software engineering conference and symposium on the foundations of software engineering.*

Honnibal, M., & Montani, I. (2017). *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.*

Hossain, M. Z., Sohel, F., Shiratuddin, M. F., & Laga, H. (2019). A Comprehensive Survey of Deep Learning for Image Captioning. *Association for Computing Machinery*.

Hu, R., Fried, D., Rohrbach, A., Klein, D., Darrell, T., & Saenko, K. (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.*

ISO. (2012). *Robots and robotic devices* (Tech. Rep.). International Organization for Standardization.

Jo, J., & Bengio, Y. (2017). Measuring the tendency of CNNs to learn surface statistical regularities. *ArXiv*.

Ke, L., Li, X., Bisk, Y., Holtzman, A., Gan, Z., Liu, J., … Srinivasa, S. (2019). Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., … Farhadi, A. (2017). AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*.

Liddy, E. D. (2001). Natural language processing.

Liu, C., Zhu, F., Chang, X., Liang, X., Ge, Z., & Shen, Y.-D. (2021). Vision-language navigation with random environmental mixup. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).*

Ma, C.-Y., Lu, J., Wu, Z., AlRegib, G., Kira, Z., Socher, R., & Xiong, C. (2019). Self-monitoring navigation agent via auxiliary progress estimation. In *Proceedings of the International Conference on Learning Representations (ICLR).*

Ma, C.-Y., Wu, Z., AlRegib, G., Xiong, C., & Kira, Z. (2019). The regretful agent: Heuristic-aided navigation through progress estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Majumdar, A., Shrivastava, A., Lee, S., Anderson, P., Parikh, D., & Batra, D. (2020). Improving vision-and-language navigation with image-text pairs from the web. In *Proceedings of the European Conference on Computer Vision (ECCV).*

Manterola, R. (2021). *Enhanced vision-language navigation by using scene recognition auxiliary task.*

Marcus, G. (2018). Deep learning: A critical appraisal. *ArXiv*.

MordorIntelligence. (2021). *Global Service Robotics Market - Growth, Trends, COVID-19 Impact, and Forecasts (2022 - 2027)* (Tech. Rep.).

Qi, Y., Pan, Z., Hong, Y., Yang, M., van den Hengel, A., & Wu, Q. (2021). The road to know-where: An object-and-room informed sequential bert for indoor vision-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).*

Qi, Y., Wu, Q., Anderson, P., Wang, X., Wang, W. Y., Shen, C., & van den Hengel, A. (2020). REVERIE: Remote Embodied Visual Referring Expression in Real Indoor Environments. *arXiv*.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., ... Fox, D. (2020). ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. *arXiv*.

Szeliski, R. (2011). *Computer vision algorithms and applications.*

Tan, H., & Bansal, M. (2019). Lxmert: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 conference on empirical methods in natural language processing.*

Tan, H., Yu, L., & Bansal, M. (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.*

Wang, H., Wang, W., Liang, W., Xiong, C., & Shen, J. (2021). Structured scene memory for vision-language navigation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Wang, H., Wang, W., Shu, T., Liang, W., & Shen, J. (2020). Active visual information gathering for vision-language navigation. In *Proceedings of the European Conference on Computer Vision (ECCV).*

Wu, W., Chang, T., & Li, X. (2021). Visual-and-language navigation: A survey and taxonomy. *ArXiv*.

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. In *Proceedings of the International Conference on Learning Representations (ICLR).*

Zhao, M., Anderson, P., Jain, V., Wang, S., Ku, A., Baldridge, J., & Ie, E. (2021). On the evaluation of vision-and-language navigation instructions. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume.*

Zhu, F., Zhu, Y., Chang, X., & Liang, X. (2020). Vision-language navigation with self-supervised auxiliary reasoning tasks. In *Proceedings of the IEEE Conference on Computer*

Vision and Pattern Recognition (CVPR).

APPENDIX

A. MATTERPORT3DMETADATA PARSER

A.1. Regions

We map region labels represented with a letter through the dictionary shown in Figure A.1

```
LABEL_MAPPING = {
    'a': 'bathroom',
    'b': 'bedroom',
    'c': 'closet',
    'd': 'dining room',
    'e': 'entryway',
    'f': 'family room',
    'g': 'garage',
    'h': 'hallway',
    'i': 'library',
    'j': 'laundry',
    'k': 'kitchen',
    'l': 'living room',
    'm': 'conference room',
    'n': 'lounge',
    'o': 'office',
    'p': 'terrace',
    'r': 'game room',
    's': 'stairs',
    't': 'toilet',
    'u': 'utility room',
    'v': 'tv',
    'w': 'gym',
    'x': 'outdoor area',
    'y': 'balcony',
    'z': 'other room',
    'B': 'bar',
    'C': 'classroom',
    'D': 'dining booth',
    'S': 'spa',
    'Z': 'junk',
    '-': 'other room'
}
```

Figure A.1. Environment descriptions used to map regions letter labels.

A.2. Objects

We map object names represented with an index though the table provided by Matterport [1]. Sampled rows are shown in Figure A.2.

[1] https://github.com/niessner/Matterport/blob/master/metadata/category_mapping.tsv

index	raw_category	category
1	wall	wall
2	door	door
3	ceiling	ceiling
4	floor	floor
5	picture	picture
6	window	window
7	chair	chair
8	doorframe	door frame
9	remove	remove
10	pillow	pillow
11	object	object
12	light	light
13	cabinet	cabinet
14	curtain	curtain
15	table	table
16	plant	plant
17	decoration	decoration
18	window frame	window frame
19	lamp	lamp
20	mirror	mirror
21	towel	towel
22	sink	sink
23	shelf	shelf

Figure A.2. Objects names used to map objects labels indexes.

We parse metadata columns and generate others that are easier to use for ranking objects importance and then for creating crafted instructions.

Distance: Objects metadata includes `px, px, pz` positions. We then calculate distance to the object through equation A.1.

$$distance = \sqrt{{p_x}^2 + {p_y}^2} \tag{A.1}$$

Elevation: We calculate where in the vertical view the object is located through equation A.2.

$$elevation = \arctan(p_z, distance) \tag{A.2}$$

Heading: We calculate heading where the object is located through equation A.3.

$$heading = \frac{\pi}{2} - \arctan(p_x, p_y) \tag{A.3}$$

Distance between node and object: We calculate the distance between nodes and objects in order to rank the best objects for describing the path to a target node. Having the node and object position and heading relative to the current node, we use equation A.4. Distances are illustrated in Figure A.4.

$$\begin{aligned} distance\_between^2 = {} & object\_distance^2 + viewpoint\_distance^2 \\ & - 2 \cdot object\_distance \cdot viewpoint\_distance \\ & \cdot \cos\left(current\_heading - object\_heading\right) \end{aligned} \tag{A.4}$$

Figure A.3. Distance between a node and an object calculated relative to the current node.

Bounding box: For visualizing, we calculate the x, y positions of the box and then the width and height. We use the object heading, elevation and oriented bounding box. M is an adjustable constant.

$$bboxV_x = \left(\frac{object\_heading}{2\pi} + 0.5\right) \cdot IMG\_WIDTH - r_1 \cdot M \tag{A.5}$$

$$bboxV_y = \left(0.5 - \frac{object\_elevation}{\pi}\right) \cdot IMG\_HEIGHT - r_0 \cdot M \tag{A.6}$$

$$bboxWidth = r_1 \cdot 2M \tag{A.7}$$

$$bboxHeight = r_0 \cdot 2M \tag{A.8}$$

B. CRAFTED AND GENERATED INSTRUCTIONS

B.1. Crafted instructions

We generate crafted instructions using objects retrieved from `360-visualization` our script based on Matterport metadata. Several steps are performed for generating, which are described below. Firstly, we filter the retrieved objects for each view, in order to use them for generating atomic instructions. We aim to concatenate these atomic instructions and generate the crafted instruction.

For filtering, we evaluate the objects and their information in different conditions:

- **Vision cone**: We keep the objects that are within the cone of vision towards the next node, this is, (-$\pi/2$, +$\pi/2$). This way, we only reference objects that help us describe the next action.
- **Distance**: We only use objects that are closer than the target node. Most of the detected objects on the metadata are far away and are not useful as a reference for describing the path to the next node.

- **Elevation**: We filter all objects that are up to 0.6 on elevation, because we don't want to include roof objects, like lamps, lights, among others.
- **Vocabulary**: We filter objects that are not on the original dictionary.
- **Noisy objects**: We filter objects like roof, window, floor, and other names that are present on all environments.

Secondly, we rank all objects, considering different metrics:

- **Size**: We have the object dimensions, so we rank their importance on the scene by their size. We weigh the size by their distance.
- **Uniqueness**: More repeated objects have lower ranking that more unique objects.

Having the best object for each view, we can use them for creating atomic instructions. These are based on rules, which are described below.

With the next node within the vision range ($-\pi/2$, $\pi/2$)

- **Move to the same room with object**: If the next node is in the same room as the current one, and at least one object is detected. We get the orientation of the object respect to the path and choose randomly one of the next atomic instructions:
  - Go straight with the {`OBJECT_NAME`} on your {`OBJECT_ORIENTATION`}
  - Walk straight down the {`opposite_direction(OBJECT_ORIENTATION)`} side of the {`OBJECT_NAME`}
- **Move to the same room without object**: If the next node is in the same room as the current one, but there are no objects detected. We get the orientation of the next node and select randomly one of the next atomic instructions:
  - Turn a little to the {`NEXT_NODE_ORIENTATION`} and walk forward.
  - Walk straight a little to the {`NEXT_NODE_ORIENTATION`}.
  - Walk straight slightly to the {`NEXT_NODE_ORIENTATION`}.

  - Turn slightly to the {NEXT_NODE_ORIENTATION} and walk forward.
- **Move to other room with object**: If the next node is in other room and at least one object is detected. We get the current room name, the next room name, the object name and the object orientation. We randomly choose one of the next atomic instructions:
  - Exit the {CURRENT_REGION} to the {NEXT_REGION} walking by the {opposite_orientation(OBJECT_ORIENTATION)} side of the {OBJECT_NAME
  - Go out of {CURRENT_REGION} into the {NEXT_REGION} walking with the {(OBJECT_NAME)} on your {OBJECT_ORIENTATION}.
- **Move to other room without object**: If the next node is in other room and no objects are detected. We get the current room name and the next room name. We use the next atomic instruction:
  - Exit the {CURRENT_REGION} to the {NEXT_REGION}.

With the next node out of the vision range ($-\pi/2$, $\pi/2$)

We first need to rotate, in order to have the next node inside the vision cone. We randomly choose one of below instructions:

- Turn {HARD_ROTATION_ORIENTATION}
- Make a {HARD_ROTATION_ORIENTATION}
- Take a {HARD_ROTATION_ORIENTATION}

with the exception of "turn around" that we use separately. Turned to the vision cone inside ($-\pi/2$, $\pi/2$), we then concatenate this instruction with the string obtained with above conditions, generating only one instruction for each action.

We also add an stop phrase at the end, for orientating the agent where to stop. Having the last node, we find the closer object on the scene. With this object we apply the rules presented below:

- **Object is on front of the target node**: We randomly choose one of these:
  - Stop in front of the {`OBJECT_NAME`}
  - Wait just in the front of {`OBJECT_NAME`}
- **Object is on behind the target node**: We randomly choose one of these:
  - Wait with the {`OBJECT_NAME`} `on your back`
  - Stop passing by the {`OBJECT_NAME`} `behind you`
- **Object is left or right**: We randomly choose one of these:
  - Wait at the {`opposite_orientation(OBJECT_ORIENTATION)`} of the {`OBJECT_NAME`}
  - Stop on the {`opposite_orientation(OBJECT_ORIENTATION)`} of the {`OBJECT_NAME`}

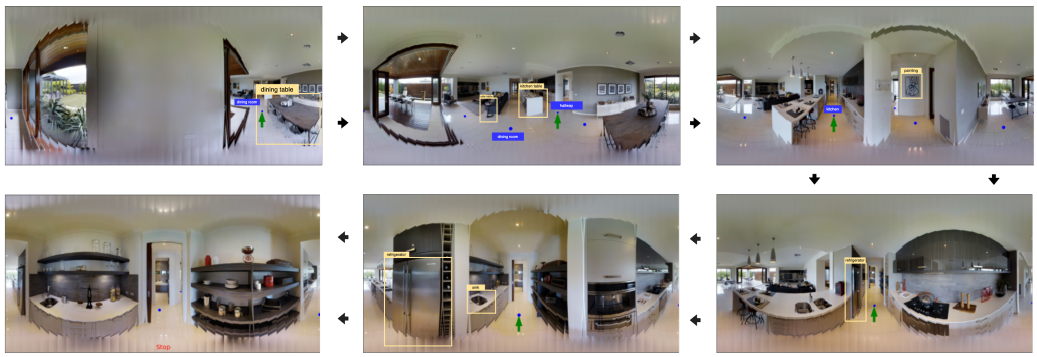
For each node, we create an atomic instruction referencing the next action. We then concatenate all these atomic instructions with the final (stop) atomic instruction and create our crafted instruction. Extra examples are shown on Figures B.1 and B.2

B.2. Generated instruction from baseline module with auxiliary tasks

Generated instructions from baseline model including auxiliary tasks: objects auxiliary task and crafted instructions auxiliary task. Example is shown on B.3

C. PRE-TRAINING THE REGRETFUL AGENT WITH GENERATED INSTRUCTIONS

We pre-train the Regretful-Agent with instructions generated from the Speaker module trained with auxiliary tasks. The best two configurations of these auxiliary tasks were $\lambda = 0.5$ or $\beta = 0.3$. Graphs of Success Rate weighted by Path Length (SPL) and Distance from Goal are shown in Figures C.1 and C.2.



- Walk passed the dining table and continue towards the kitchen. Walk through the kitchen passed the counters and stove and stop near the sink.

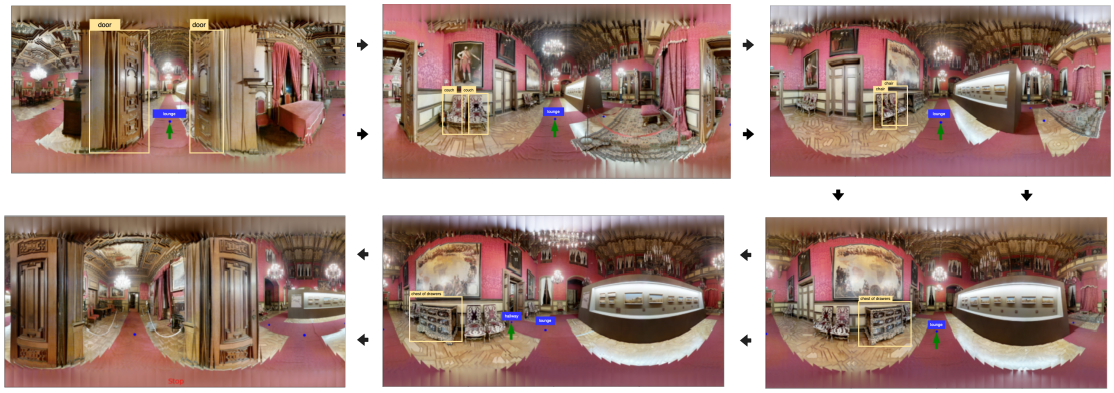
- Move to the far left corner of the table. Proceed to the doorway to your right.
walk into the kitchen with the stove on your right. proceed forward until the fridge is on your left
proceed forward once more until there is a small sink on your left and shelves on your right.

- Walk straight past the table and turn right to go between the kitchen counters and walk straight past the refrigerator into the pantry and stop halfway between the two shelves on the right.

- Make a right, walk straight down the left side of the dining table. exit the dining room to the hallway walking by the left side of the side table. exit the hallway to the kitchen. walk straight down the right side of the refrigerator. go straight with the sink on your left. wait at the right of the faucet

Panoramic views sequence above and human instructions + crafted instruction below. Images are sequenced through the arrows.

Figure B.1. First crafted instructions example.



- Follow the red carpet straight and make the first left where the red carpet continues to go through the wooden doors. Wait outside of the room before entering through the wooden doors.

- Exit through double doors, walk along the red carpet to the other end of the room. Wait by the second set of double doors.

- Walk down the hallway and pass all the picture frames placed on the right until the end. Turn left to enter another room, and stop there.

- Exit the hallway to the lounge. walk straight down the right side of the chair. walk forward. go straight with the chest of drawers on your left. exit the lounge to the hallway

Panoramic views sequence above and human instructions + crafted instruction below. Images are sequenced through the arrows.

Figure B.2. Second crafted instructions example.

- Walk forward and stop at the doormat in front of the door. Turn left and continue walking. Stop at the bottom of the stairs. Walk to the top of the stairs. Then turn left and walk up the stairs. Stop walking when you're halfway up the stairs.

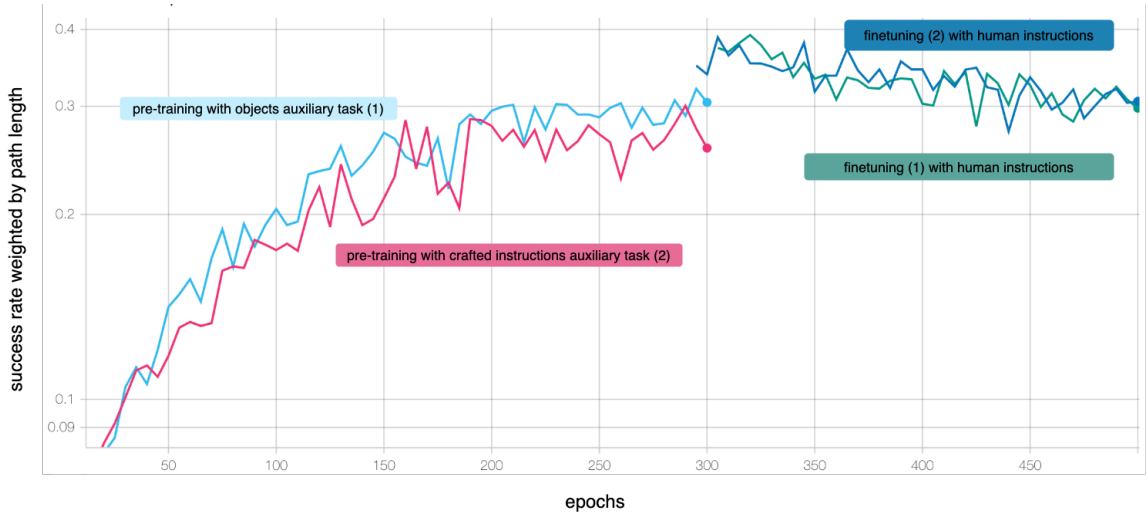
- Walk straight and turn left. Walk up the stairs. Stop on the second step from the top.

- Walk through the open door and then turn left. Walk up the stairs . stop on the landing at the end of the next set of stairs.

- Walk past the bathroom and turn left. Walk up the stairs. Stop on the second step from the top.

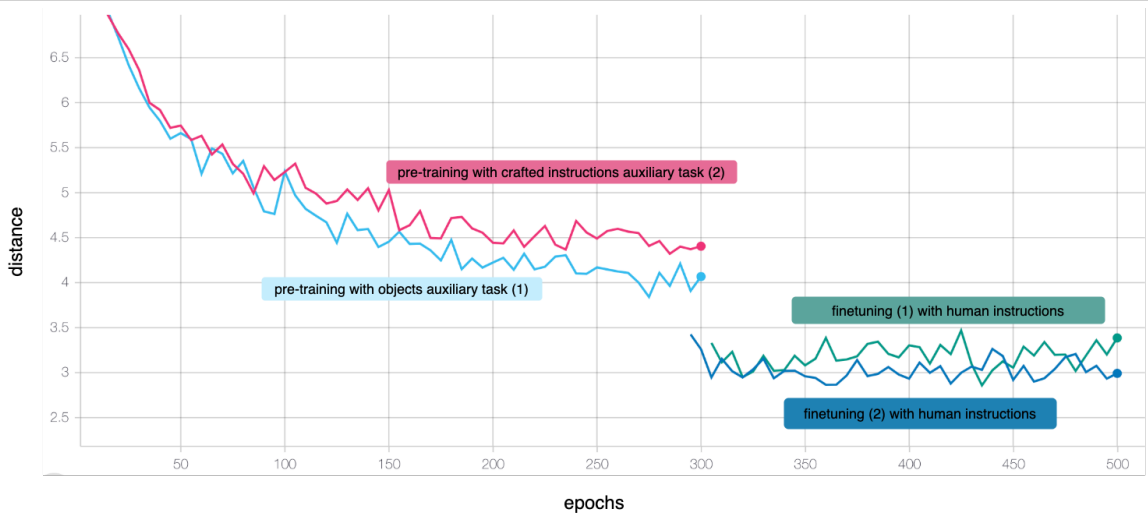
Panoramic views sequence above and human instruction + speaker follower instructions + speaker follower with objects auxiliary task + speaker follower with crafted instructions auxiliary task below. Images are sequenced through the arrows.

Figure B.3. Baseline module with auxiliary tasks output example.



300 epochs of pre-training with generated instructions and 200 epochs for finetuning with human instructions.

Figure C.1. Graph of success rate weighted by path length through epochs.



300 epochs of pre-training with generated instructions and 200 epochs for finetuning with human instructions.

Figure C.2. Graph of distance from goal through epochs.