Bridging the Visual Semantic Gap in VLN via Semantically Richer Instructions

Joaquín Ossandón(✉), Benjamín Earle, and Álvaro Soto

Pontificia Universidad Católica de Chile, Santiago, Chile
{jiossandon,biearle}@uc.cl, asoto@ing.puc.cl

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-the-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 DL 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, demonstrating the advantages of the proposed data augmentation method.

Keywords: Computer vision · Natural language processing · Navigation · VLN · Data augmentation

1 Introduction

The ability of a robot to receive an instruction in natural language and navigate in unknown environments has been an attractive research topic in recent years [7,15,16,22,24,28]. In particular, the Visual-and-Language Navigation (VLN) task [1] proposes that an agent can follow a textual instruction such as "*Go up*

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-19836-6_4.

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 is still out of reach for autonomous robots. As an example, under current benchmarks [25], state-of-the-art AI models based on Deep Learning (DL) do not reach the intended goal position more than 65% of the time [12].

There are several reasons that can help to explain the low performance of current models to face the VLN task [25]. 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 [17]. As a consequence, current VLN models exhibit limited generalization capabilities, leading to a large drop in performance when they are tested in unseen environments [7,22,25].

In effect, today there is abundant experimental evidence indicating that current DL based models operate as associative memory engines triggered by superficial data correlations [2,3,10], fostering the detection of direct stimulus-response associations. Indeed, given enough parameters, DL models are able to memorize arbitrary noisy data [26]. 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 to 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 [1] that are generated with an improved version of the Speaker-Follower model presented in [7]. 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, after publication, we will 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 [4] 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. Section 2 describes the VLN task and current benchmarks. Section 3 reviews relevant previous works. Section 4 presents an experimental setup to highlight the limitations of current VLN models to use visual information. Afterwards, Sect. 5 describes the construction of our visual semantically richer instructions for the VLN task. Finally, Sect. 6 presents our conclusions and future research avenues.

2 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 lack of real images in the proposed problems [1]. In 2017, the Matterport3D dataset [4] was introduced, containing RGB-D building scale scenes of 90 different home environments. Later that year, a new navigation problem was proposed: Room-to-Room (R2R) [1], 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, a total of 21,567 navigation instructions with an average of 29 words [1].

We construct over Matterport3DMeta a set of tools named `360-visualization`[2] for getting objects and navigable nodes with their intrinsic data for each view, as shown in Fig. 1.

[1] https://github.com/niessner/Matterport/tree/master/metadata.
[2] https://github.com/cacosandon/360-visualization.

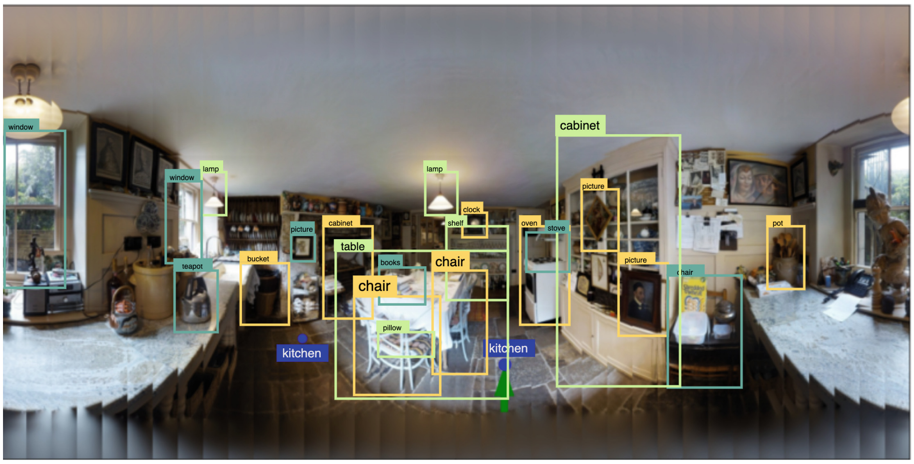


Fig. 1. Objects and viewpoints visualization sampled with `360-visualization` scripts.

2.1 The Visual-and-Language Navigation Task

The task of VLN for an agent is to follow natural language instructions from an initial to a target position through navigation in a real environment, simulated by Matterport3D Simulator [1]. 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 a 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 $\langle a_0, a_1, .., a_T \rangle$ 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`> action, and the task is successful if the agent arrives at a location near the target position, recognizing it as the goal.

3 Related Work

VLN task has been the main motivation for many researchers on Computer Vision. Interesting surveys and reviews [9,25] talk about several techniques developed over the baseline architecture proposed by R2R.

They group them on categories such as the inclusion of auxiliary tasks [13,16,19,21,28], the improvement of navigation and exploration [11,13,14,23,24] and curriculum learning with data augmentation [7,15,22].

Data augmentation has become an essential part of training in various tasks, not only increasing quantity of training data, but also providing more informative data to reduce overfitting, and improve generalization and performance

[6,18,20]. On navigation, different approaches have proposed augmenting training instructions [7,22] but it has been shown that they do not follow human syntax or include relevant information [27].

That's why we focus on this topic, basing our study on the Speaker-Follower [7], which consists of two modules: one that follows instructions (follower) and other that performs data augmentation to feed the training of the follower (speaker), which we improve for generate new semantically richer instructions.

State-of-art leaderboard is summarized in Table 1. VLN-BERT+REM [12] has the highest success rate (SR), followed by SSM [23] and Active Gathering [24]. These models have high overhead costs due to the time and resources required by their complex architectures. We demonstrate that focusing on data augmentation greatly benefits navigation performance without making models even more complex.

Table 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 [7]	14.82	6.62	0.28	0.35
Tactical Rewind [11]	22.08	5.14	0.41	0.54
Self-Monitoring [13]	17.11	5.99	0.32	0.43
Environmental Dropout [22]	11.70	–	0.47	0.51
Regretful-Agent [14]	13.69	5.69	0.40	0.48
ORIST [19]	10.90	4.72	0.51	0.57
VLN-BERT + REM [12]	13.11	3.87	0.59	**0.65**
SSM [23]	20.7	4.32	0.45	0.62
Active Gathering [24]	20.6	4.36	0.4	0.58

4 Models Problem

Aiming to demonstrate models deficits, we experiment in both visual and linguistic areas on the state-of-the-art models, based on [9] analysis. A summary diagram is shown in Fig. 2.

4.1 Visual Area

In order to evaluate the effectiveness of visual components within VLN architectures, it is interesting to know the importance of visual scene information when deciding the next action on navigation through the environment.

We run our experiments over Self-Monitoring [13] and Regretful-Agent [14], because of their public codebase[3,4]. Each of these architectures were trained in

[3] https://github.com/chihyaoma/selfmonitoring-agent/.
[4] https://github.com/chihyaoma/Regretful-Agent.

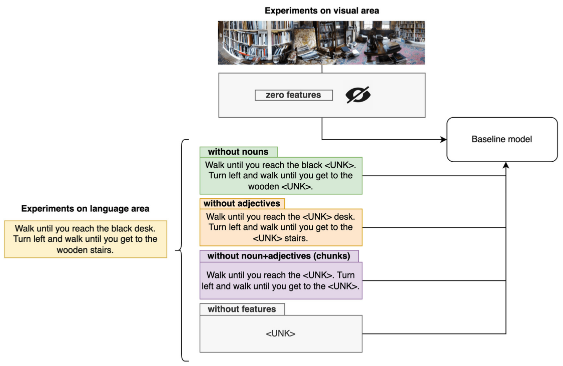

Experiments on visual area
zero features
Baseline model
without nouns
Walk until you reach the black <UNK>. Turn left and walk until you get to the wooden <UNK>.
without adjectives
Walk until you reach the <UNK> desk. Turn left and walk until you get to the <UNK> stairs.
without noun+adjectives (chunks)
Walk until you reach the <UNK>. Turn left and walk until you get to the <UNK>.
without features
<UNK>
Experiments on language area
Walk until you reach the black desk. Turn left and walk until you get to the wooden stairs.

Fig. 2. Experiments diagram. Visual features are replaced with zeros. In the linguistic area, four experiments are performed: without nouns, without adjectives, without nouns+adjectives and without features.

two different conditions. The first condition is 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 178,000 sampled routes with associated instructions generated with the Speaker module [7], the same instructions we improve later in this work. Six experiments evaluated in known (seen) and unknown (unseen) environments were performed, which are shown in Tables 2 and 3.

4.2 Language Area

We also experiment changing the text of the instruction, in order to check which components are relevant for the agent to better decide.

We use spaCy [8], an NLP model used in the industry to obtain text features. Each word of each instruction was classified according to the context, as adjective, noun or other. The Regretful-Agent model was trained by extracting from each instruction: all adjectives, all nouns, all nouns+adjectives and extracting the whole text (i.e. without linguistic features), training the model for a total of 100 h with the 8 experiments, which results are shown in Table 4.

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 [1]. We also use a new metric called success rate weighted by Path Length (SPL) [13,15,22,28], that measures the success rate normalized by path length.

Table 2. Visual ablation study on Self-Monitoring [13] and Regretful-Agent [14], seen environment with Single Run (not Beam Search).

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`	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`	19.05	7.6	0.14	0.27

Table 3. Visual ablation study on Self-Monitoring [13] and Regretful-Agent [14], unseen environment with Single Run (not Beam Search).

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`	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`	18.8	6.62	0.19	0.36

Table 4. Language ablation study on Self-Monitoring [13] and Regretful-Agent [14], unseen environment with Single Run (not Beam Search). w/means without

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

4.3 Ablation Studies

When experimenting in the visual area by removing all the visual features we notice a clear difference between seen and unseen environments. In seen environments the difference in success rate is very large (Table 2). 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 both models without pre-training don't improve by more than 7% SR over the blind model. This demonstrates good memorization but lack of generalization, being visual information almost useless on previously unknown scenes.

When experimenting in the linguistic area, we noticed that when we extract the whole language, 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 point 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.

Because of the high-level instructions, removing adjectives increases the SR, indicating that nouns descriptions are actually interfering with the model performance.

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.

5 Semantically Richer Instructions Proposal

To navigate using vision, we must first learn to follow semantically meaningful instructions that foster the use of visual information. We create simple instructions that are scene-object based, referencing them and their context, enriching over generic non visual instructions like "go straight".

We construct our model over Speaker-Follower, but using only the Speaker module. On Fig. 3 is 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 pass them across an encoder module. This encoder gives us an encoded context `ctx`, 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.

On R2R, each path has three instructions. For each, the Speaker module builds the loss as the Negative Log Loss (NLL) between the corresponding word of the instruction and the generated word, as shown on the figure.

Generated instructions on the Speaker module are now being used for almost all state-of-the-art models of VLN task on a pre-training phase. However, it has

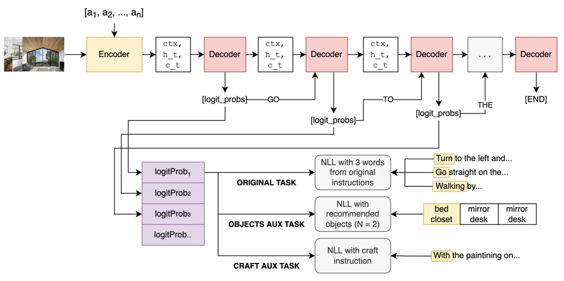


Fig. 3. Speaker with proposed auxiliary tasks.

been shown that they do not follow human syntax, they have orientation problems and do not include relevant information, being incorrect in most cases [27].

Using Matterport3DMeta, we propose to add relevant objects to generated instructions applying two loss auxiliary tasks: objects and crafted instructions, aiming the vision to be mandatory for the agent to navigate. These two are also included in Fig. 3.

5.1 Objects and Crafted Instructions

Object metadata is available on Matterport3DMeta, but it is raw and difficult to use. That's why we created `360-visualization`, a script for fetching and visualizing objects and navigable viewpoints on each node, for each heading and elevation.

These objects are the main component of the objects auxiliary task, but we also use them for the generation of crafted instructions.

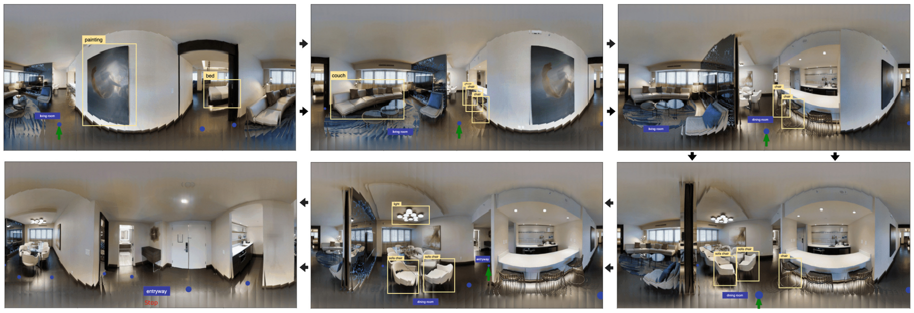
For a specific path, we generate an atomic instruction for each node on the sequence. Having the current 360 ° visual image and the next node we can select the best object to reference, following a set of rules. For instance, in Fig. 4 we start with a big painting at the right of the next node, generating 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.

5.2 Objects Auxiliary Task

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

We assign the best `N` objects to each word of the instructions of that path, matching word index with the closer node index.

For instance, in Fig. 3 we recommend the model to use "bed" and "closet" (`N = 2`) for the first word.



```
- Walk past the large picture and chair. Walk past the dining room table turn right into the hallway and stop.
```

```
- 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.
```

```
- With the painting of the mermaid towards your right, head straight. After passing the counter towards your
right, turn right and wait.
```

```
- 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.
```

Fig. 4. Crafted instruction example. Panoramic views sequence on top and human instructions + crafted instruction on bottom. Images are sequenced through the arrows. Presented objects and scenes names are sampled from the data.

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 on training the Speaker results as follows

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

A resulting generated instruction with Speaker + objects auxiliary task is shown on Fig. 5.

5.3 Crafted Instructions Auxiliary Task

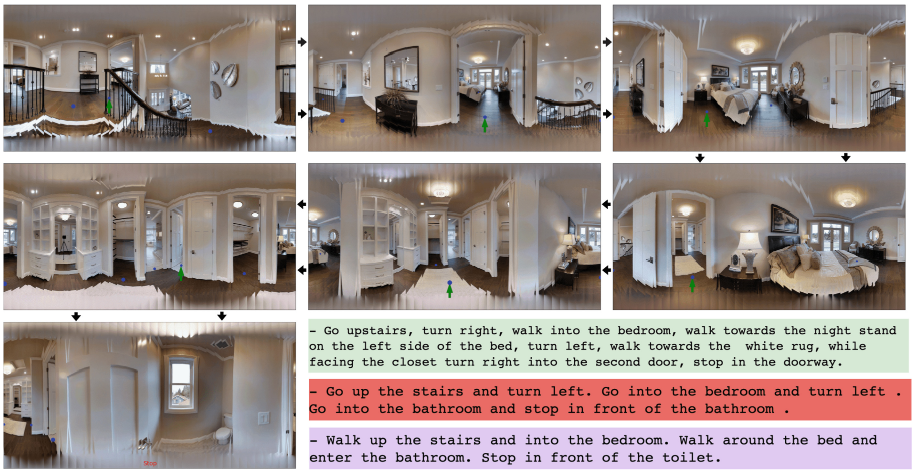
Having our own crafted instructions, we use them directly on training and also adding a Negative Log Loss between the generated instruction and the crafted instruction, word by word.

The sum of these losses are weighted by β, another modifiable parameter. The final loss for each generated word is

$$wordLoss = \sum_{i=0}^{3} NLL\left(log(logit), w_{original_i}\right) + \beta \cdot NLL\left(log(logit), w_{crafted}\right)$$

5.4 Results and Discussion

After training the Speaker module with auxiliary tasks, we generate instructions based on the same sampled paths as the original augmented dataset with



- Go upstairs, turn right, walk into the bedroom, walk towards the night stand on the left side of the bed, turn left, walk towards the white rug, while facing the closet turn right into the second door, stop in the doorway.

- Go up the stairs and turn left. Go into the bedroom and turn left . Go into the bathroom and stop in front of the bathroom .

- Walk up the stairs and into the bedroom. Walk around the bed and enter the bathroom. Stop in front of the toilet.

Fig. 5. Objects auxiliary task instruction example. Panoramic views sequence above and human instruction + base speaker instruction + new generated instruction below. Images are sequenced through the arrows.

different values for λ and β, hyperparameters that weigh auxiliary tasks loss. We execute separated auxiliary tasks, because it is redundant information. Crafted instructions used in the crafted instructions auxiliary task use the same objects as the ones we pass directly on the objects auxiliary task.

We then train the Regretful Agent navigation model with the augmented data or crafted instructions directly (Phase I), and finetune it with the original training data (Phase II). Results for seen and unseen environments are on Tables 5 and 6.

Best results are achieved pre-training with instructions generated using objects auxiliary task with $\lambda = 0.5$ or crafted instructions auxiliary task with $\beta = 0.3$, increasing up to 51% the success rate on unseen environment. Using our new augmented instructions the success rate increment is doubled compared using original Speaker module synthetic instructions, this is, +8% versus the baseline model.

We extend the experiments to the HAMT model [5], pre-training with the best settings we got from above. Results are on Table 7. Although we get a lower increase, adding visual information to the instructions is still useful across all models. As future work, we can test with different configurations of the auxiliary tasks to see which dataset generates higher benefits. We make objects and crafted instructions available in `360-visualization`[5] so they can be added to any language model, as we propose with the Speaker module.

The new module with auxiliary tasks have several advantages in the quality of the generated instructions. One example is shown on Fig. 5.

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

First, it corrects the Speaker module in orientation, since we help the model by indicating which objects to reference in the instruction, relating them with the next node position. For instance, in the figure's sequence the agent must turn right, while the Speaker generates an instruction that wrongly says the opposite.

Second, compared to human instructions we realize that a complex instruction is not necessary, since agent is a beginner on navigation. Humans describe the path in a high level, which makes it even more complicated to extract visual information. Our module generates low-level instructions, describing the best way to orientate another DL model.

At last, generated instructions reference objects that exist in the environment but not in the original instructions, as it does with the word "toilet" at the end of the instruction showed in Fig. 5. The model learned to use objects even though it has never seen them before, nor has any type of information (such as the instruction showed on the figure, which is from the validation set, and therefore, never seen before).

We also demonstrate that the increase of success rate using original data augmentation does not depend on the quality of the Speaker's instructions, but rather the quantity is the main contributor. This means, quantity compensates quality for improving performance (178,300 augmented instructions over 4675 train instructions).

As we mentioned before, using the same 178,300 sampled paths, we also create totally new crafted instructions (Fig. 4) without the need of human instructions in order to use them as pre-training. On Tables 5 and 6 it is shown as "`PWIF Crafted Directly`". We almost reach the same success rate as the full training (Phases I + II) with human instructions and we exceed base training by 4%.

Table 5. Regretful-Agent [14] pre-training with instructions generated from different models, evaluated on **seen** environments with Single Run (not Beam Search) and original human instructions. `PWIF` means *Pre-training with instructions from* and `AT` means *Auxiliary Task*.

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

Table 6. Regretful-Agent [14] pre-training with instructions generated from different models, evaluated on **unseen** environments with Single Run (not Beam Search) and original human instructions. `PWIF` means *Pre-training with instructions from* and `AT` means *Auxiliary Task*.

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

Table 7. HAMT [5] pre-training with instructions generated from best Speaker configurations (ranked by success rate after pre-training the Regretful Agent), evaluated on **unseen** environments with Single Run (not Beam Search) and original human instructions. `PWIF` means *Pre-training with instructions from* and `AT` means *Auxiliary Task*.

HAMT +	SPL ↑	SR ↑
`Without pre-training`	54.4	48.7
`PWIF Speaker base`	56.3	52.3
`PWIF Speaker + Objects AT` $\lambda = 0.5, N = 2$	**57.4**	52.4
`PWIF Speaker + Crafted AT` $\beta = 0.3$	57.3	**52.6**

We then show that the Speaker module as a instruction generator does not contribute more than our crafted instructions generated based on rules, unless we add the proposed auxiliary tasks, where the performance increase is remarkable.

6 Conclusions

Different methodologies have been developed to improve scene understanding, in order to achieve a better performance in navigation with human interaction. They focus mainly on model architecture, leaving aside the base of the task: the dataset. As we present, navigation agents do not use visual information available on environments for making a decision. Removing visual features generates a slight success rate drop of only 7% on unseen environments, evidencing that the

R2R dataset has instructions that do not reference the context in which agent is situated. This last factor allows agents to execute actions in almost a random manner, reaching the goal anyway.

In addition, these same instructions are too complex and high level, confusing agents that start as beginners on navigation. To bridge the visual semantic gap presented on the datasets, we create new semantically richer instructions.

For this purpose, we use scene objects and crafted instructions to feed a set of auxiliary tasks. The resulting model generates new instructions that help to correct the errors existing in the original instructions, while increase the success rate by 8% in unseen environments when we use them as pre-training, which doubles the increase of the original Speaker. We then demonstrate that the creation of semantically richer instructions that include explicit visual information allows the agent to better learn to navigate.

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

- **Own object detection**: 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.

Acknowledgments. This work was partially funded by FONDECYT grant 1221425, the National Center for Artificial Intelligence CENIA FB210017, Basal ANID and by ANID through *Beca de Magister Nacional* N° 22210030.

References

1. Anderson, P., et al.: Vision-and-language navigation: interpreting visually-grounded navigation instructions in real environments. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018)
2. Arpit, D., et al.: A closer look at memorization in deep networks. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70 (2017)
3. Belkin, M., Hsu, D.J., Mitra, P.: Overfitting or perfect fitting? Risk bounds for classification and regression rules that interpolate. In: Advances in Neural Information Processing Systems (2018)
4. Chang, A., et al.: Matterport3D: learning from RGB-D data in indoor environments. In: International Conference on 3D Vision (3DV) (2017)

5. Chen, S., Guhur, P.L., Schmid, C., Laptev, I.: History aware multimodal transformer for vision-and-language navigation. In: NeurIPS (2021)
6. Feng, S.Y., et al.: A survey of data augmentation approaches for NLP. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021 (2021)
7. Fried, D., et al.: Speaker-follower models for vision-and-language navigation. In: Neural Information Processing Systems (NeurIPS) (2018)
8. Honnibal, M., Montani, I.: spaCy 2: natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing (2017, to appear)
9. Hu, R., Fried, D., Rohrbach, A., Klein, D., Darrell, T., Saenko, K.: 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 (2019)
10. Jo, J., Bengio, Y.: Measuring the tendency of CNNs to learn surface statistical regularities. arXiv (2017)
11. Ke, L., et al.: Tactical rewind: self-correction via backtracking in vision-and-language navigation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
12. Liu, C., Zhu, F., Chang, X., Liang, X., Ge, Z., Shen, Y.D.: Vision-language navigation with random environmental mixup. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2021)
13. Ma, C.Y., et al.: Self-monitoring navigation agent via auxiliary progress estimation. In: Proceedings of the International Conference on Learning Representations (ICLR) (2019)
14. Ma, C.Y., Wu, Z., AlRegib, G., Xiong, C., Kira, Z.: The regretful agent: heuristic-aided navigation through progress estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
15. Majumdar, A., Shrivastava, A., Lee, S., Anderson, P., Parikh, D., Batra, D.: Improving vision-and-language navigation with image-text pairs from the web. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12351, pp. 259–274. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58539-6_16
16. Manterola, R.: Enhanced vision-language navigation by using scene recognition auxiliary task (2021)
17. Marcus, G.: Deep learning: a critical appraisal. arXiv (2018)
18. Mikołajczyk, A., Grochowski, M.: Data augmentation for improving deep learning in image classification problem. In: 2018 International Interdisciplinary PhD Workshop (IIPhDW) (2018)
19. Qi, Y., et al.: The road to know-where: an object-and-room informed sequential BERT for indoor vision-language navigation. In: ICCV (2021)
20. Shorten, C., Khoshgoftaar, T.: A survey on image data augmentation for deep learning. J. Big Data (2019)
21. Tan, H., Bansal, M.: LXMERT: learning cross-modality encoder representations from transformers. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (2019)
22. Tan, H., Yu, L., Bansal, M.: 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 (2019)

23. Wang, H., Wang, W., Liang, W., Xiong, C., Shen, J.: Structured scene memory for vision-language navigation. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
24. Wang, H., Wang, W., Shu, T., Liang, W., Shen, J.: Active visual information gathering for vision-language navigation. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12367, pp. 307–322. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58542-6_19
25. Wu, W., Chang, T., Li, X.: Visual-and-language navigation: a survey and taxonomy. arXiv (2021)
26. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: ICLR (2017)
27. Zhao, M., et al.: 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 (2021)
28. Zhu, F., Zhu, Y., Chang, X., Liang, X.: Vision-language navigation with self-supervised auxiliary reasoning tasks. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)