VELMA: Verbalization Embodiment of LLM Agents for Vision and Language Navigation in Street View

Raphael Schumann[1], **Wanrong Zhu**[2], **Weixi Feng**[2], **Tsu-Jui Fu**[2],
Stefan Riezler[1,3], **William Yang Wang**[2]
[1] Computational Linguistics, Heidelberg University, Germany
[3] IWR, Heidelberg University, Germany
[2] University of California, Santa Barbara
{rschuman, riezler}@cl.uni-heidelberg.de
{wanrongzhu, weixifeng, tsu-juifu, william}@ucsb.edu

Abstract

Incremental decision making in real-world environments is one of the most challenging tasks in embodied artificial intelligence. One particularly demanding scenario is Vision and Language Navigation (VLN) which requires visual and natural language understanding as well as spatial and temporal reasoning capabilities. The embodied agent needs to ground its understanding of navigation instructions in observations of a real-world environment like Street View. Despite the impressive results of LLMs in other research areas, it is an ongoing problem of how to best connect them with an interactive visual environment. In this work, we propose VELMA[1], an embodied LLM agent that uses a verbalization of the trajectory and of visual environment observations as contextual prompt for the next action. Visual information is verbalized by a pipeline that extracts landmarks from the human written navigation instructions and uses CLIP to determine their visibility in the current panorama view. We show that VELMA is able to successfully follow navigation instructions in Street View with only two in-context examples. We further finetune the LLM agent on a few thousand examples and achieve 25%-30% relative improvement in task completion over the previous state-of-the-art for two datasets.

1 Introduction

Large language models (LLMs), which have shown impressive reasoning capabilities in traditional natural language processing tasks, are increasingly used as the reasoning engine of embodied agents for, e.g., household robots [25], video games [30] and indoor navigation [35]. These tasks are mostly based on simulations that either feature computer-generated images with a fixed set of displayable objects and textures, or are limited in scale and trajectory length. In this paper, we present a verbalization embodiment of an LLM agent (VELMA) for urban vision and language navigation in Street View. The unique challenge of this task is the combination of a large-scale environment derived from an actual road network, real-world panorama images with dense street scenes, and long navigation trajectories. The agent needs to ground its understanding of the navigation instructions in the observable environment and reason about the next action to reach the target location. The navigation instructions are written by humans and include open-ended landmark references and directional indications intended to guide the agent along the desired path. In order to leverage the reasoning capabilities of LLMs, we use embodiment by verbalization, a workflow where the task, including the agent's trajectory and visual observations of the environment, is verbalized, thus

[1]Project page: https://velma.schumann.pub

Foundation Models for Decision Making Workshop at NeurIPS 2023.

Navigate to the described target location!
Action Space: forward, left, right, turn_around, stop
Navigation Instructions:
"*Orientate yourself such that* ***a blue bench*** *is on your right, go to the end of the block and make a right. Follow the* ***park*** *on your left and make a right at the next intersection. Pass the* ***black fire hydrant*** *on your right and stop when you get to the* ***gray door on the brown building****.*"
Action Sequence:

There is a blue bench on your left.
1. **turn_around**
There is a blue bench on your right.
2. **forward**
3. **forward**
4. **forward**
There is a 3-way intersection.
5. **right**
6. **forward**
There is a park on your left.
7. **forward**
There is a park on your left.
8. **forward**
There is a 4-way intersection.
9. *<next word prediction>*

Figure 1: Prompt sequence used to utilize LLMs for VLN in Street View. Verbalized observations of the visual environment are in green and appended to the prompt at each step. Agent actions (blue) are acquired by LLM next word prediction. Highlighting of text for visual presentation only. Full navigation trajectories are, on average, 40 steps long.

embodying the LLM via natural language. Figure 1 shows the verbalization at the ninth step of the current trajectory for a given navigation instance. At each step, the LLM is prompted with the current text sequence in order to predict the next action. Then the predicted action is executed in the environment, and the new observations are verbalized and appended to the prompt. This is repeated until the agent eventually predicts to stop.

The main contributions of our work are as follows: (i) We introduce VELMA, to our knowledge, the first LLM-based agent for urban VLN. (ii) We report few-shot results for the urban VLN task and achieve new state-of-the-art performance by finetuning our agent on the training set. (iii) We address and resolve limitations of the commonly used Touchdown environment [4], making it amenable for few-shot agents.

2 Related Work

Outdoor VLN Agent models for the outdoor/urban VLN task [4] commonly follow a sequence-to-sequence architecture where encoded text and image representations are fused for each decoder step [32, 15, 23, 26]. Other proposed agents employ pretrained vision and language transformers that are finetuned on task-specific data [38, 2]. Zhong et al. [34] represent the visual environment by symbols using semantic segmentation and extreme downsampling of panorama images, but their agent does not improve over previous success rates. Other work uses CLIP to score the presence of extracted landmarks at each panorama node in a graph and uses this information to plan a route for given navigation instructions [24]. Their non-urban environment has a graph with 300 nodes, and the navigation path is planned a priori with full access to all panorama images and landmark scores. In contrast, our agent is embodied and has to plan ad-hoc with access to directly observed information only.

Indoor VLN Indoor agents [8, 31, 27, 9, 37, 17, 10, 5, 14] are used for navigation datasets like R2R [1] and RxR [13] or ObjectNav [19, 36]. Recently, Zhou et al. [35] introduced an LLM-based agent for R2R that incorporates image information by transcribing its entire content with an image-to-text model. This is feasible because the navigation trajectories are only six steps on average compared to 40 steps in the urban VLN task considered in our work. Another notable indoor VLN agent uses CLIP to directly predict the next action by scoring the compatibility of the current sub-instruction with available waypoint images [7].

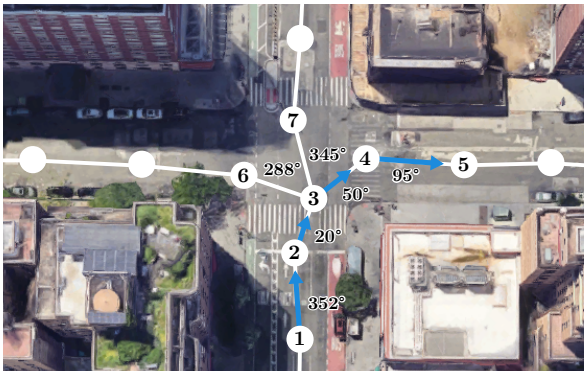


Figure 2: The Touchdown environment introduced by Chen et al. [4] can require action sequences that are semantically inconsistent with the correct navigation instructions. In the depicted subgraph, the action sequence to move from node 1 to node 5 is to move FORWARD four times. The semantically correct sequence of actions would include a right turn in between. We fix the ambiguity by modifying the environment and selecting the desired direction at intersections in relation to all outgoing streets.

3 Urban VLN Environment

We use the Touchdown environment introduced by Chen et al. [4]. The environment is based on Google's Street View and features 29,641 full-circle panorama images connected by a navigation graph. It covers the dense urban street network spanning lower Manhattan. The navigation graph is a directed graph $G = \langle V, E \rangle$ where each edge $\langle v, v' \rangle \in E$ is associated with $\alpha_{\langle v,v' \rangle}$ which is the heading direction from node v to node v' ranging from 0° to 360°. The agent state $s = (v, \alpha)$ is composed of its current position $v \in V$ and its heading direction α. The agent can move by executing an action $a \in \{\texttt{FORWARD}, \texttt{LEFT}, \texttt{RIGHT}, \texttt{STOP}\}$. The state transition function $s_{t+1} = \phi(a_t, s_t)$ defines the behavior of the agent executing an action. In Chen et al. [4], the agent's heading α_t at position v is restricted to align with the heading of an outgoing edge $\alpha_{\langle v,v' \rangle}$. In case of the RIGHT action, the new state s_{t+1} is $(v, \alpha_{\langle v,\vec{v} \rangle})$ where $\vec{v}$ is the neighboring node closest to the right of the agent's current heading. In other words, the agent is rotated in place to the right until it *snaps* to the direction of an outgoing edge. Likewise, for the LEFT action. In the case of the FORWARD action, the agent moves along the edge $\langle v, v' \rangle$ according to its current heading direction $\alpha_t = \alpha_{\langle v,v' \rangle}$. The environment is then forced to automatically rotate the agent's heading towards an outgoing edge: $\alpha_{t+1} = \alpha_{\langle v',v^* \rangle}$ where v^* is the neighbor node in the direction closest to the previous heading α_t.

3.1 Alignment Inconsistencies in Touchdown

As described in Schumann and Riezler [23], the automatic rotation mentioned above can lead to generalization problems, e.g., when moving towards the flat side of a T-intersection. For example, if the agent is automatically rotated towards the right facing street and subsequently executes the RIGHT action, it rotates towards the direction it came from instead of clearing the intersection in the intended direction. The same problem also occurs at intersections with more than three directions. Figure 2 gives an illustrative example that shows the navigation graph at a 4-way intersection. Because the environment is derived from a real-world street layout, the nodes in the graph are not perfectly arranged as in an artificial grid world. In order to make a right turn at the intersection and to follow the route from v^1 to v^5, one expects to use the action sequence [FORWARD, FORWARD, RIGHT, FORWARD, FORWARD]. However, when the agent reaches v^3, it is automatically rotated towards the closest outgoing edge, in this case, $\langle v^3, v^4 \rangle$. This is because the rotation 20°→50° towards v_4 is shorter than the rotation 20°→345° towards v_7. As such, the required sequence of actions to go from v^1 to v^5 in Chen et al. [4]'s environment is [FORWARD, FORWARD, FORWARD, FORWARD]. This is unpredictable and is not correctly aligned with "*turn right at the intersection*" instructions.[2] To alleviate this problem, Schumann and Riezler [23] explicitly feed the change of heading at each timestep as additional input to their model. This enables the agent to anticipate the unexpected rotation and to adapt to it. Because adding heading delta values to the text-based interface makes it convoluted and unnecessarily difficult for few-shot learning, we propose a more intuitive way to solve this ambiguity at intersections. We modify the state transition function ϕ such that the agent is not automatically rotated when moving FORWARD. This means the agent's heading α_t is not automatically aligned with an outgoing edge. Instead, the direction is selected in relation to all outgoing edges. The agent at node v^3 in Figure 2 has the nodes v^6, v^7 and v^4 in front. The forward direction is selected as the middle one of the three edges, the right direction as the right-most edge, and the left direction as the left-most edge.

[2] In the Appendix we show more examples for 3-way, 4-way and 5-way intersections.

This means that executing the RIGHT action at position v^3 will now rotate the agent towards node v^4 and allows to use the semantically correct sequence of actions for the depicted route. The proposed modification solves the issue of inconsistent action sequences at intersections and allows to use agents that are not specifically trained in this environment.

3.2 Turning Around

We additionally introduce the TURN_AROUND action which lets the agent reverse its direction: $s_{t+1} = (v, \alpha_t - 180°)$. In the unmodified environment, this is achieved using the LEFT or RIGHT action on regular street segments. The new action is better aligned with natural language verbalizations of direction reversal and promotes intuitive communication with the environment.

4 Navigation Task

The objective of the navigation task is to find the goal location by following the given navigation instructions. A navigation instance is defined by the initial state s_1, target node $\hat{v}_T$, gold path $(\hat{v}_1, \hat{v}_2 ..., \hat{v}_T)$ and navigation instructions text $n = (w_1, w_2, ..., w_N)$. The agent starts at s_1 and predicts the next action a_1 based on the navigation instructions and current observations. These are the panorama image and number of outgoing edges at the current position. The environment processes the action and puts the agent into a new state: $s_2 = \phi(a_1, s_1)$. This is repeated until the agent predicts STOP at the presumed goal location. If the agent stops within one neighboring node of the target node, the navigation objective is considered accomplished.

4.1 Challenges

One main challenge to successfully follow the navigation instructions is to reliably detect landmarks in the panorama images along the route. The landmarks mentioned in the instructions are open-ended and can refer to any object or structure found in street scenes, including vegetation, building features, vehicle types, street signs, construction utilities, company logos and store names. The agent also needs to posses different types of reasoning, most importantly spatial reasoning to follow general directions, locate landmarks and evaluate stopping conditions. In addition, the agent needs to understand the temporal aspect of the task and reason about the sequence of previous observations and actions. See Table 1 for example snippets from the navigation instructions.

Egocentric Spatial Reasoning	
1.	... turn so the orange construction barrier is on your left ...
2.	... a red truck in front of you ...
3.	... a playground on the far right corner ahead ...
Allocentric Spatial Reasoning	
4.	... green metal pole with pink flowers on top ...
5.	... building with columns around the windows ...
6.	... stop in between Chase and Dunkin' Donuts ...
Temporal Reasoning	
7.	... go straight until you see Chipotle and then ...
8.	... once you passed the underpass ...
9.	... stop when the park on your right ends ...
Other	
10.	... proceed straight through three more intersections ...
11.	... you should see TD Bank on your left ...
12.	... if you see Dory Oyster Bar, you have gone too far ...

Table 1: Reasoning skills the embodied LLM agent must possess in order to successfully complete the navigation task. Each with three example snippets from the navigation instructions.

4.2 Datasets

There are two datasets that provide navigation instructions for the environment described in Section 3: **Touchdown** [4] and **Map2seq** [22]. Each dataset includes around 10k navigation instances, and we utilize them in the more challenging *unseen* scenario introduced by Schumann and Riezler [23]. This means that generalization is crucial because the training routes are located in an area that is geographically separated from the area of development and test routes. The main difference between the two datasets is that Touchdown instructions were written by annotators who followed the route in Street View, while Map2seq instructions were written by annotators that saw a map of the route. The Map2seq navigation instructions were later validated to also be correct in Street View. Another difference is that the initial state in Map2seq orientates the agent towards the correct direction which leads to overall better task completion rates than for Touchdown instances.

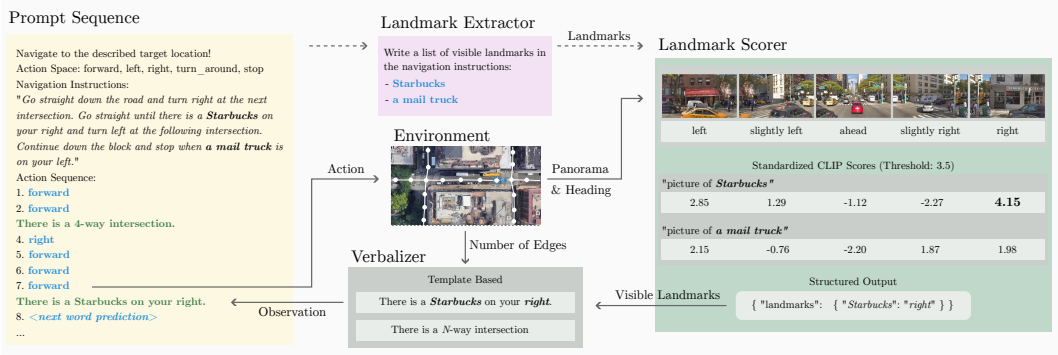


Figure 3: Overview of the proposed agent VELMA navigating in the Street View environment. The prompt sequence includes the task description, navigation instructions, and verbalized navigation trajectory up to the current timestep. The next action is decided by next word prediction utilizing an LLM and subsequently executed in the environment. This puts the agent into a new state, and the landmark scorer determines if an extracted landmark is visible in the current panorama view. The verbalizer takes this landmark information along with the information about a potential intersection and produces the current observations text. This text is then appended to the prompt sequence and again used to predict the next action. This process is repeated until the agent stops and the alleged target location.

5 LLM Agent

In this section, we propose the urban VLN agent that uses an LLM to reason about the next action. To this end, we verbalize the navigation task, especially the environment observations. The workflow includes the extraction of landmarks that are mentioned in the instructions and determining their visibility in the current panorama image. The verbalizer then integrates the visible landmarks and street intersections into an observation text phrase o_t at each step. The complete text prompt at timestep t is composed as follows:

$$x_t = [d^a, n, d^b, o_1, 1, a_1, o_2, 2, a_2, ..., o_t, t], \tag{1}$$

where [ ] denotes string concatenation, d^a and d^b are part of the task description and n is the navigation instructions text. Punctuation and formatting are omitted in the notation for brevity. Figure 3 shows a prompt sequence at $t = 8$ on the left. This formulation of the navigation task enables the agent to predict the next action by next word prediction:

$$a_t = \arg\max_{w \in A} P_{LLM}(w|x_t), \tag{2}$$

where A are the literals of the five defined actions and P_{LLM} is a black-box language model with no vision capabilities.

5.1 Landmark Extractor

Each navigation instructions text n mentions multiple landmarks for visual guidance. In order to determine if a mentioned landmark is visible in the current panorama view, we first have to extract them from the instructions text. For this, we create a single prompt that includes five in-context examples of navigation instructions paired with a list of landmarks. It is used by the LLM to generate the list of landmarks $(l_1, l_2, ..., l_L)$ mentioned in the given navigation instructions. The landmark extractor is depicted in the top middle of Figure 3 and is applied once, before the navigation starts.

5.2 Landmark Scorer

At each step, the agent observes a panorama view p_v^α, defined by its current position v and heading direction α. The view is an 800x460 sized image cut from the panorama with 60° field-of-view. In order to determine if a landmark l_i is visible in the view, we employ a CLIP model [18] to embed the image and the caption: "*picture of* $[l_i]$". The similarity score of the two embeddings determines the

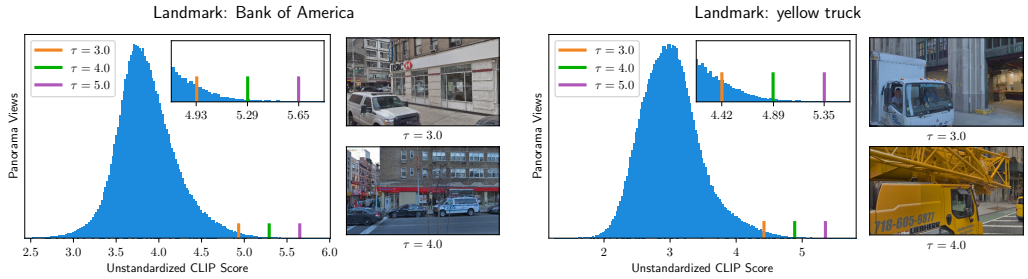


Figure 4: Distribution of CLIP scores between a landmark and panorama images in the training area. The CLIP score represents the semantic similarity of the panorama image and the text caption "picture of [landmark]". The distribution is used to standardize the score of the landmark and a novel panorama. The threshold τ is defined on the standardized score and used to determine the visibility of the landmark in the novel panorama image.

visibility of the landmark. Because the scores can be biased towards certain types of landmarks, we standardize them using all views p^*_{train} of the ~20k panorama images in the training area. Recall that we operate in the *unseen scenario* where the training area and evaluation area are geographically separated. The standardized score of a landmark is:

$$z(l, p_v^\alpha) = \frac{\text{CLIP}(l, p_v^\alpha) - \mu(C_l)}{\sigma(C_l)} \quad \text{where } C_l = \{\text{CLIP}(l, p_{v'}^{\alpha'}) \mid p_{v'}^{\alpha'} \in p^*_{train}\}. \tag{3}$$

If the standardized score is larger than the threshold τ, the landmark is classified as visible in the current view. There are no ground truth labels indicating whether landmarks are visible in certain panoramas, thus the classification is completely unsupervised. The threshold is the only tunable parameter in the landmark scorer. Figure 4 shows the distribution of unstandardized CLIP scores and views at different threshold values for two example landmarks. While the views at $\tau = 4.0$ both show the correct landmark, the view at $\tau = 3.0$ for "Bank of America" shows an HSBC branch, and for "yellow truck" it shows a white truck. This suggests that the optimal threshold lies between the two values. As depicted on the right in Figure 3, the agent also evaluates views to the left and right of the current heading. Each panorama view direction $(p_v^{\alpha-90^\circ}, p_v^{\alpha-45^\circ}, p_v^\alpha, p_v^{\alpha+45^\circ}, p_v^{\alpha+90^\circ})$ is associated with a string literal m valued *left*, *slightly left*, *ahead*, *slightly right* or *right*, respectively. A visible landmark l_i and the corresponding direction literal m_i are passed to the verbalizer. A full navigation trajectory includes around 200 image views (40 steps and 5 view directions per step) and each landmark is typically visible in only one or two views.

5.3 Verbalizer

The verbalizer is a template-based component that produces environment observations in text form. There are two types of environment observations. First, there are street intersections that are detected based on the number of outgoing edges $N(v)$ at the current node v in the navigation graph. If there are three or more outgoing edges at step t, the verbalizer encodes this information into the observation string o_t^e: "*There is a* $[N(v)]$*-way intersection*". Extracting this information directly from the navigation graph is akin to the junction type embedding used by the ORAR model [23] and is motivated by direction arrows displayed in the Street View GUI that human navigators used during data collection. The other type of observations are landmarks visible in the panorama view. The landmark name l_i and direction literal m_i are used to verbalize the observation o_t^l: "*There is* $[l_i]$ *on your* $[m_i]$". The complete observation is $o_t = [o_t^e, o_t^l]$, where the respective string is empty if no intersection or landmark is detected. The observation is appended to the prompt in Equation 1 and used by the agent to decide the next action.

6 Experiments

We conducted experiments[3] to evaluate the navigation performance of the proposed LLM agent in finetuning and in-context learning settings. We ran the landmark extractor once for all instances using GPT-3 [3] and used the same extracted landmarks in all experiments. Manual inspection shows that GPT-3 reliably extracts all landmarks without obvious errors. We used `CLIP-ViT-bigG-14-laion2B-39B-b160k` [21] as the CLIP model in the landmark scorer. We set the threshold $\tau = 3.5$ for all experiments. The threshold was selected by inspecting the distribution of CLIP scores (as in Figure 4) for a handful of landmarks. On purpose, we did not systematically tune it in order to not violate the premise of few-shot learning.

6.1 Metrics and Baseline

We use three metrics to measure navigation performance. The task completion (TC) rate is a binary metric that measures whether the agent successfully stopped within one neighboring node of the target location. Shortest-path distance (SPD) calculates the shortest path length between the stopping location and goal location [4]. Key point accuracy (KPA) measures the ratio of correct decisions at key points. Key points include the initial step, intersections along the gold route, and the target location.

For baselines, we use the current state-of-the-art agent model for urban VLN called ORAR [23]. The model employs a seq-to-seq architecture where the encoder LSTM reads the navigation instructions text, and the multi-layer decoder LSTM receives image feature vectors of the current panorama view as additional input at each action decoding step. The ORAR model is a very strong baseline beating more sophisticated models like the VLN Transformer [38]. Because the environment modifications introduced in Section 3 spare the agents from learning specific irregularities, we additionally retrain ORAR in the improved environment for a fair comparison.

6.2 Few-Shot Learning Results

The proposed text-only interface allows us to use large language models as reasoners without updating their weights or fusing image representations. The prompt consists of a short task description and two in-context examples (2-shot). The examples are full text sequences for randomly selected navigation instances in the training set. The two plots in Figure 5 show that performance scales with parameter count and varies across model families. The `FORWARD`-baseline reveals that OPT [33] can barely compete with a baseline that yields nonsensical predictions, even at a model size of 65 billion parameters. LLaMa [28] and especially LLaMa-2 [29] show promising navigation skills reaching 48.3 and 57.7 key point accuracy (KPA) on Touchdown and Map2seq, respectively. However, this KPA score only translates to task completion (TC) rates of 2.1 and 3.2, revealing that the model is not able to consistently predict correct actions throughout the whole navigation trajectory. The only few-shot LLMs that achieve substantial TC rates are GPT-3 and GPT-4 [16]. As listed in Table 2, VELMA-GPT-4 achieves the best results for the 2-shot setting. It reaches 44% and 77% of the TC rate reported for the previous state-of-the-art model ORAR♠ [23] which is a seq-to-seq model that has direct access to image features and was trained on the full training set. In contrast, the GPT models act as a blind agent that solely relies on observation descriptions produced by the verbalizer. This is remarkable because LLMs are not explicitly trained to experience embodiment

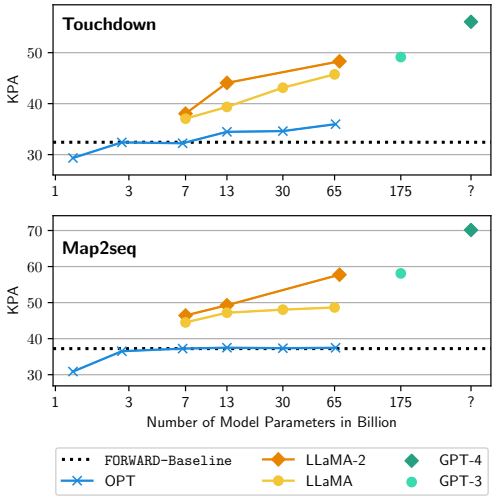


Figure 5: Key point accuracy (KPA) for 2-shot in-context learning of large language models with increasing parameter count. The `FORWARD`-Baseline predicts only walking forward and is better than random actions.

[3]Code is provided in the supplementary material and will be released upon acceptance.

Model	Development Set						Test Set					
	Touchdown			Map2seq			Touchdown			Map2seq		
	SPD↓	KPA↑	TC↑	SPD↓	KPA↑	TC↑	SPD↓	KPA↑	TC↑	SPD↓	KPA↑	TC↑
Seq-to-Seq RNN, full training set												
ORAR	20.0 ±0.7	-	15.4 ±2.2	11.9 ±0.4	-	27.6 ±1.8	20.8 ±0.6	-	14.9 ±1.2	13.0 ±0.3	-	30.3 ±1.8
ORAR♠	16.5 ±0.1	**64.0** ±0.2	22.6 ±0.6	10.3 ±0.4	74.4 ±0.8	29.9 ±1.7	17.4 ±0.2	**62.3** ±0.1	19.1 ±1.0	10.9 ±0.1	74.7 ±0.2	32.5 ±1.4
2-Shot In-Context Learning												
VELMA-GPT-3	22.2	49.1	6.8	19.1	58.1	9.2	-	-	-	-	-	-
VELMA-GPT-4	<u>21.8</u>	<u>56.1</u>	<u>10.0</u>	<u>12.8</u>	<u>70.1</u>	<u>23.1</u>	-	-	-	-	-	-
LLM Finetuning, full training set												
VELMA-FT	18.6 ±0.3	61.3 ±0.6	22.6 ±0.4	9.8 ±0.5	76.8 ±0.8	38.2 ±0.5	19.4 ±0.1	60.0 ±0.4	21.0 ±0.2	10.3 ±0.4	76.4 ±0.6	37.8 ±0.9
VELMA-RBL	**16.1** ±0.1	63.5 ±0.1	**25.7** ±0.4	**8.4** ±0.7	**78.9** ±0.8	**42.8** ±0.6	**16.6** ±0.2	61.9 ±0.2	**24.1** ±0.7	**8.8** ±0.2	**77.9** ±0.2	**42.1** ±1.1

Table 2: Results for the urban VLN task on Touchdown and Map2seq. ORAR is the previous best model and follows a seq-to-seq architecture that processes text and image features. ORAR♠ and following models make use of the improved environment introduced in Section 3. The proposed environment and workflow allow to perform the urban VLN task as text-only next word prediction. VELMA-GPT-3 and VELMA-GPT-4 models are prompted with two in-context examples. Due to cost and data leakage concerns, we evaluate the GPT models on the development sets only. VELMA-FT is based on LLaMa-7b that is finetuned on all ~ 6k training text sequences. VELMA-RBL training is described in Section 6.3.1. All results are for the *unseen scenario* where evaluation routes are geographically separated from training routes. Experiments are repeated three times with different random seeds (mean/std reported). **Bold** values are the nominal best results and <u>underlined</u> are best few-shot results.

in a visual environment. This is emergent behavior unearthed by verbalizing the VLN task. We also observe that the GPT models use the `TURN_AROUND` action in appropriate ways, even when the in-context examples do not demonstrate the usage of it. This emphasizes the effectiveness of intuitive communication with the environment.

6.3 Finetuning Results

To further explore the capabilities of the proposed LLM agent, we finetune LLaMa-7b on all training instances of the respective dataset, denoted by VELMA-FT in Table 2. Each training instance is the full text sequence after following the gold path. The visibility of landmarks is determined by the landmark scorer during training because gold annotations are not available. There are 6,770 training instances for Touchdown and 5,737 for Map2seq. We finetune for 20 epochs using LoRA [11] to adapt attention query and value projections. The best model is selected by task completion on the development set. The resulting agent outperforms the previous state-of-the-art model ORAR♠ by 10% and 16% relative TC rate. Comparing ORAR♠ which fuses image features at the vector level to VELMA-FT which finetunes on verbalizations of observations, shows that the text-based environment observations are less prone to overfitting.

6.3.1 Response-Based Learning

A navigation task is successfully completed if the agent stops at either the goal location or an adjacent neighboring node. Training the agent with teacher-forcing to exactly follow the gold route penalizes the agent for stopping one step short or one step past the target node, despite accomplishing the navigation objective. Furthermore, the agent can not learn to recover from incorrect decisions during inference. We thus train the agent to directly optimize the TC metric while also feeding it its own actions during training, called VELMA-RBL in Table 2. The procedure for VELMA-RBL is inspired by response-based learning [6] and imitation learning [20] and is outlined in Algorithm 1. The loss for an instance at training step j is either computed by teacher forcing the gold action sequence $\hat{\mathbf{a}}$, or by student forcing, determined by a mixing parameter λ. In the latter case, the agent freely moves around, executing the actions decoded using its current model weights θ_j. If the agent successfully stops within one neighboring node of the target location, the predicted action sequence $\mathbf{a}_j$ is considered correct and used as the reference to train the agent. If the agent stops at the wrong location, an oracle path is computed to provide the optimal counterfactual action at each step in the trajectory. In our case, the oracle's optimal next action is computed as the shortest path to the goal location. We set

Algorithm 1 RBL Optimization of Task Completion

Require: mixing ratio λ, training step j, model weights θ_j, gold action sequence $\hat{\mathbf{a}}$, prompt x_1
  if $random(0,1) < \lambda$ **then**
    $\mathbf{a}_{\theta_j} = StudentForcing(\theta_j, x_1)$
    $\mathbf{a}_j = \arg\max \mathbf{a}_{\theta_j}$
    if $TaskCompletion(\mathbf{a}_j) = 1$ **then**
      $loss_j = \mathcal{L}_{CE}(\mathbf{a}_{\theta_j}, \mathbf{a}_j)$
    else
      $\mathbf{a}^*_j = Oracle_{stepwise}(\mathbf{a}_j)$
      $loss_j = \mathcal{L}_{CE}(\mathbf{a}_{\theta_j}, \mathbf{a}^*_j)$
    end if
  else
    $\mathbf{a}_{\theta_j} = TeacherForcing(\theta_j, x_1, \hat{\mathbf{a}})$
    $loss_j = \mathcal{L}_{CE}(\mathbf{a}_{\theta_j}, \hat{\mathbf{a}})$
  end if

$\lambda = 0.5$ to collect training losses in a batch evenly from both training strategies. Manually inspecting trajectories produced by the trained agent, we found improvements of following instructions that have stopping criteria like "*Stop a few steps before Y.*" or "*Stop at X. If you see Y you have gone too far.*". In both cases, the agent learned to walk past the uncertain stopping location and to invoke the `TURN_AROUND` action in order to walk back once landmark Y appeared. Overall the task completion rate increases significantly using this training procedure with 15% and 11% relative improvements for Touchdown and Map2seq respectively. Overall, our contributions in this work amount to an absolute increase of task completion by 9.2 and 11.8 over the previously reported state-of-the-art.

6.4 Image Ablation

In this section, we ablate the image model used by the landmark scorer. We finetune a LLaMa-7b model according to Section 6.3 and use CLIP [18], OpenCLIP [21] or no image model in the landmark scorer. The latter case means that no landmark observation is passed to the prompt sequence. The results in Table 3 show that OpenCLIP is better suited for detecting landmarks in our navigation task than the original CLIP model. This is in line with better ImageNet results reported by the OpenCLIP authors and suggests that the agent can directly benefit from further improvements of CLIP models. Appending no landmarks to the prompt sequence further degrades performance, especially on Touchdown.

	Touchdown		Map2seq	
Image Model	**SPD↓**	**TC↑**	**SPD↓**	**TC↑**
no image	27.4 ±0.5	14.7 ±0.5	9.7 ±0.2	35.2 ±0.9
CLIP	21.3 ±0.5	19.5 ±0.6	**9.8** ±0.3	37.2 ±0.5
OpenCLIP	**18.6** ±0.3	**22.6** ±0.4	**9.8** ±0.5	**38.2** ±0.5

Table 3: Vision ablation on the development set. We finetune a separate LLaMa-7b model for each ablation. CLIP refers to `clip-vit-large-patch14` [18]. The OpenCLIP image model refers to `CLIP-ViT-bigG-14-laion2B-39B-b160k` [21].

7 Conclusion

We introduced VELMA, an agent for urban vision and language navigation, which utilizes a large language model to infer its next action. The LLM is continuously queried with a text prompt that verbalizes the task description, navigation instructions, visual observations, and past trajectory of the agent. In order to include observed landmarks in the prompt, we propose an unsupervised pipeline that extracts landmarks from the instructions and determines their visibility in the current panorama view based on thresholded CLIP scores. We evaluate the embodied LLM agent in a modified version of the commonly used Touchdown environment based on Street View. One proposed modification is fixing a problem at intersections that led to incorrect alignments of action sequences, and another modification adds the `TURN_AROUND` action which provides a more intuitive way to communicate with the environment. The proposed agent achieves strong few-shot in-context learning results of 10 and 23 task completion rates for Touchdown and Map2seq, respectively, and yields new state-of-the-art results of 24 and 42 task completion rates when finetuned on the full training set.

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018.

[2] Jason Armitage, Leonardo Impett, and Rico Sennrich. A priority map for vision-and-language navigation with trajectory plans and feature-location cues. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1094–1103, 2023.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

[4] Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, California, 2019.

[5] Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. *Advances in neural information processing systems*, 34:5834–5847, 2021.

[6] James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 18–27, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `https://aclanthology.org/W10-2903`.

[7] Vishnu Sashank Dorbala, Gunnar Sigurdsson, Robinson Piramuthu, Jesse Thomason, and Gaurav S. Sukhatme. Clip-nav: Using clip for zero-shot vision-and-language navigation. In *CoRL 2022 Workshop on Language and Robot Learning*, 2022. URL `https://www.amazon.science/publications/clip-nav-using-clip-for-zero-shot-vision-and-language-navigation`.

[8] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *Neural Information Processing Systems (NeurIPS)*, 2018.

[9] Tsu-Jui Fu, Xin Eric Wang, Matthew F Peterson, Scott T Grafton, Miguel P Eckstein, and William Yang Wang. Counterfactual vision-and-language navigation via adversarial path sampler. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 71–86. Springer, 2020.

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

[11] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=nZeVKeeFYf9`.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, California, 2015.

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

[14] Jialu Li, Hao Tan, and Mohit Bansal. Envedit: Environment editing for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15407–15417, 2022.

[15] Harsh Mehta, Yoav Artzi, Jason Baldridge, Eugene Ie, and Piotr Mirowski. Retouchdown: Releasing touchdown on StreetLearn as a public resource for language grounding tasks in street view. In *Proceedings of the Third International Workshop on Spatial Language Understanding (SpLU)*, Online, 2020.

[16] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[17] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9982–9991, 2020.

[18] Alec Radford, Jong Wook Kim, Chris Hallacy, A. Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[19] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL `https://openreview.net/forum?id=-v4OuqNs5P`.

[20] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, FL, USA, 2011.

[21] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL `https://openreview.net/forum?id=M3Y74vmsMcY`.

[22] Raphael Schumann and Stefan Riezler. Generating landmark navigation instructions from maps as a graph-to-text problem. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 489–502, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.41. URL `https://aclanthology.org/2021.acl-long.41`.

[23] Raphael Schumann and Stefan Riezler. Analyzing generalization of vision and language navigation to unseen outdoor areas. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7519–7532, Dublin, Ireland, May 2022. Association for Computational Linguistics. URL `https://aclanthology.org/2022.acl-long.518`.

[24] Dhruv Shah, Blazej Osinski, Brian Ichter, and Sergey Levine. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action, 2022.

[25] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. URL https://arxiv.org/abs/1912.01734.

[26] Yanjun Sun, Yue Qiu, Yoshimitsu Aoki, and Hirokatsu Kataoka. Outdoor vision-and-language navigation needs object-level alignment. *Sensors*, 23(13), 2023. ISSN 1424-8220. doi: 10.3390/s23136028. URL https://www.mdpi.com/1424-8220/23/13/6028.

[27] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2610–2621, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1268. URL https://aclanthology.org/N19-1268.

[28] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[29] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[30] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv: Arxiv-2305.16291*, 2023.

[31] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[32] Jiannan Xiang, Xin Wang, and William Yang Wang. Learning to stop: A simple yet effective approach to urban vision-language navigation. In *Findings of the Association for Computational Linguistics (ACL Findings)*, Online, 2020.

[33] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[34] Victor Zhong, Austin W. Hanjie, Sida Wang, Karthik Narasimhan, and Luke Zettlemoyer. Silg: The multi-domain symbolic interactive language grounding benchmark. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21505–21519. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/b3e3e393c77e35a4a3f3cbd1e429b5dc-Paper.pdf.

[35] Gengze Zhou, Yicong Hong, and Qi Wu. Navgpt: Explicit reasoning in vision-and-language navigation with large language models, 2023.

[36] Kaiwen Zhou, Kaizhi Zheng, Connor Pryor, Yilin Shen, Hongxia Jin, Lise Getoor, and Xin Eric Wang. Esc: Exploration with soft commonsense constraints for zero-shot object navigation. *arXiv preprint arXiv:2301.13166*, 2023.

[37] Wang Zhu, Hexiang Hu, Jiacheng Chen, Zhiwei Deng, Vihan Jain, Eugene Ie, and Fei Sha. Babywalk: Going farther in vision-and-language navigation by taking baby steps. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2539–2556, 2020.

[38] Wanrong Zhu, Xin Wang, Tsu-Jui Fu, An Yan, Pradyumna Narayana, Kazoo Sone, Sugato Basu, and William Yang Wang. Multimodal text style transfer for outdoor vision-and-language navigation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1207–1221, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.103. URL `https://aclanthology.org/2021.eacl-main.103`.

A Finetuning Details

In the finetuning experiments, we use the official LLaMA-7b weights and finetune a LoRA adapter for *q_proj* and *v_proj*. LoRA hyperparameter are set to $r = 8$, $alpha = 16$, $dropout = 0.05$ and no bias. We use Adam [12] as the optimizer with a learning rate of 0.0003, warmup ratio of 0.1 and linear decay. The batch size is 16 and we train for 20 epochs. We use greedy decoding for all experiments.

B Modified Environment

In Section 3.1 we propose modifications to the environment introduced by Chen et al. [4]. In Table 4 we give an overview of action sequences required to clear 3-way, 4-way and 5-way intersections in different directions in the original environment implementation and our modified environment. It is clear that the action sequences required in our improved environment are more intuitive and are necessary to enable few-short agents to interact with it.

C Landmark Extraction

The landmarks mentioned in the navigation instructions are extracted before the run starts. We do this by a separate prompt that we feed to GPT-3. The prompt for Map2seq instructions is shown in Figure 8 and the one for Touchdown in Figure 9. It provides five instructions texts paired with a list of extracted landmarks as in-context examples. The lists of example landmarks were compiled by hand and the same prompt is used for each instance. There are no gold annotations for extracted landmarks and as such no quantitative evaluation is possible. In Figure 10 we show landmarks extracted by GPT-3 using this prompt.

D Landmark Scorer

We show the CLIP score distribution and panorama views at certain thresholds for additional landmarks in Figure 6. Some navigation instructions refer to the flow up traffic when orientating the agent in the beginning of Touchdown instances, e.g. "Orientate yourself with against the flow of traffic...". To support this kind instructions, we score the phrases "the front view of a vehicle" and "the rear view of a vehicle" once, before the start. Whichever phrase scores higher with the initial perspective, determines if the agent is facing against the traffic or with the flow of traffic respectively. This traffic flow prediction is then provided as an environment observation string before the first step of the agent.

E Full Prompt Sequence

In Figure 7 we show a full prompt sequence for a given navigation instance. The agent predicted `STOP` at timestep 14 and thus finished the trajectory. In the depicted case the agent followed the correct route and successfully completed the navigation objective. For visualization purposes the trajectory is shortened. On average the routes in Touchdown and Map2seq require 40 steps to be completed. This also means the agents has to evaluate 200 panorama views for each navigation instance.

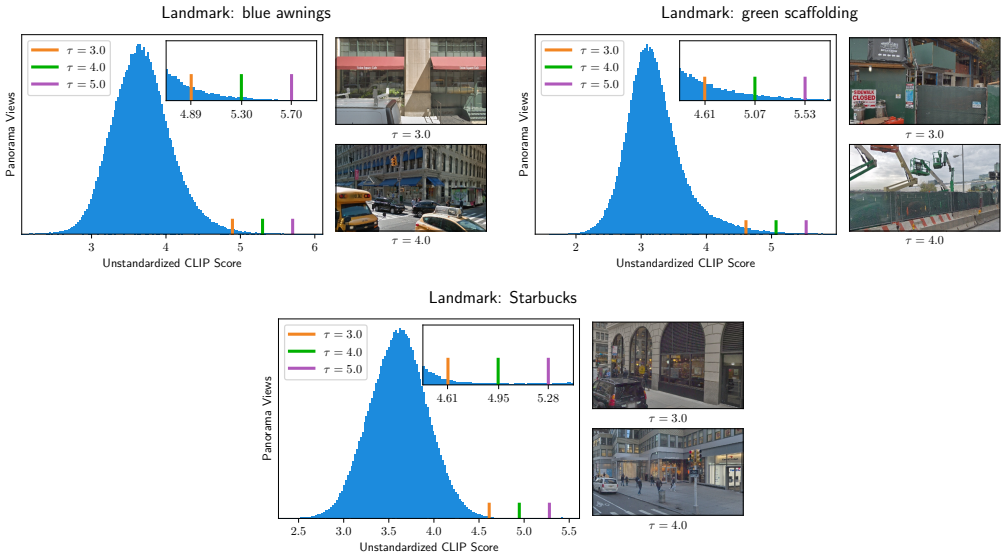


Figure 6: Distribution of CLIP scores between a landmark and panorama images in the training area. The CLIP score represents the semantic similarity of the panorama image and the text caption "picture of [landmark]". The distribution is used to standardize the score of the landmark and a novel panorama. The threshold τ is defined on the standardized score and used to determine the visibility of the landmark in the novel panorama image.

Navigate to the described target location!
Action Space: forward, left, right, turn_around, stop
Navigation Instructions:

"*Orientate yourself such that **a blue bench** is on your right, go to the end of the block and make a right. Follow the **park** on your left and make a right at the next intersection. Pass the **black fire hydrant** on your right and stop when you get to the **gray door on the brown building**.*"

Action Sequence:

There is a blue bench on your left.
1. **turn_around**
There is a blue bench on your right.
2. **forward**
3. **forward**
4. **forward**
There is a 3-way intersection.
5. **right**
6. **forward**
There is a park on your left.
7. **forward**
There is a park on your left.
8. **forward**
There is a park on your left.
9. **forward**
There is a 4-way intersection.
10. **right**
11. **forward**
There is a black fire hydrant slightly right.
12. **forward**
There is a black fire hydrant on your right.
13. **forward**
There is a gray door on the brown building on your left.
14. **stop**

Figure 7: Finished prompt sequence used to utilize LLMs for VLN in Street View. Verbalized observations of the visual environment are in green and appended to the prompt at each step. Agent actions (blue) are acquired by LLM next word prediction. Highlighting of text and shortening of route for visual presentation only. Full navigation trajectories are on average 40 steps long.

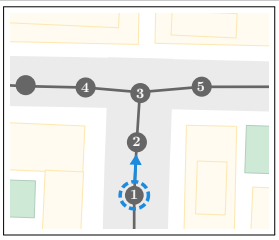
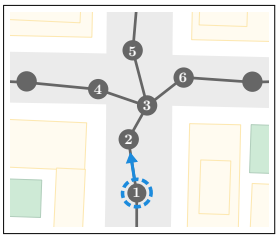
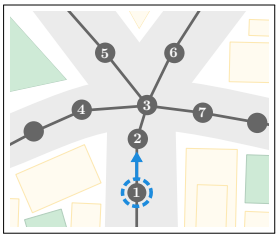
Intersection	Path	Environment by Chen et al.	Our Environment
	2→3→4 2→3→5	[FORWARD, LEFT, FORWARD] [FORWARD, FORWARD]	[FORWARD, LEFT, FORWARD] [FORWARD, RIGHT, FORWARD]
	2→3→4 2→3→5 2→3→6	[FORWARD, LEFT, LEFT, FORWARD] [FORWARD, LEFT, FORWARD] [FORWARD, FORWARD]	[FORWARD, LEFT, FORWARD] [FORWARD, FORWARD] [FORWARD, RIGHT, FORWARD]
	2→3→4 2→3→5 2→3→6 2→3→7	[FORWARD, LEFT, LEFT, FORWARD] [FORWARD, LEFT, FORWARD] [FORWARD, FORWARD] [FORWARD, RIGHT, FORWARD]	[FORWARD, LEFT, LEFT, FORWARD] [FORWARD, LEFT, FORWARD] [FORWARD, RIGHT, FORWARD] [FORWARD, RIGHT, RIGHT, FORWARD]

Table 4: Comparison of the Touchdown environment implemented by Chen et al. [4] and the improved implementation proposed by us. The action sequence required to clear an intersection in different directions in our improved environment is semantically aligned with the expected outcome.

Head to the end of the block and make a right. Pass a Subway entrance on the right and go through the light. At the next light with Staples on the corner, make a right. Stop in front of the library that is a few buildings down on the right.
Landmarks:
1. a subway entrance
2. Staples
3. a library

Go straight through the light ahead of you, then turn right at the next one. After your turn, you will see Starbucks on the left. At the light after that, turn left. Pass the church on the left and then stop after Hot Kitchen. You should be able to see a bike rental on the right.
Landmarks:
1. Starbucks
2. a church
3. Hot Kitchen
4. a bike rental

Head to the intersection and turn left. Continue to the end of the block and turn right. Go straight and past the intersection. Stop 1/3 of the way down the block with the large building on your right.
Landmarks:
None

Walk to the light with Just Sweet and turn right. Go through a light with an AMC and a couple more blocks until you see a tiny park or plaza on the far left corner. Turn left passing that park and then make a left turn almost immediately after. Stop after a couple of steps, where a road from the right joins the main road.
Landmarks:
1. Just Sweet
2. AMC
3. a park
4. a plaza

Go straight through the next 3 lights past the bus stops and at the 4th light shortly after the 3rd take a left. Stop just past the bus stop and Neta diner.
Landmarks:
1. bus stop
2. Neta diner

{*navigation instructions*}
Landmarks:
<>

Figure 8: Prompt to extract landmarks from navigation instruction in Map2seq.

You will start of at an intersection. To begin, make sure you are going in the direction of the blue and white van with orange cones around it. Pass that van. Go straight through the first intersection you get to. You will come to a light at an intersection where there is a building with a green awning. Take a right. Go straight until you are in the middle of the intersection. In front of you, there is a building with a red sign above the entrance.
Landmarks:
1. a blue and white van
2. orange cones
3. a green awning
4. a red sign above the entrance

Turn to the right until you're looking down the street. There should be a red SUV on the right side of the frame now. Begin moving forward until you reach an intersection. Take a left here. Keep moving forward until reaching a three-way intersection. Take another left here. Move forward three times. Turn to the right until you see a red and white street sign next to a series of green boards.
Landmarks:
1. a red SUV
2. a red and white street sign
3. a series of green boards

Head in the direction of traffic and continue going straight. You will have the opportunity to turn right, but DON'T. Keep going straight. When you reach the intersection, turn left. Keep going straight. You will reach an intersection, but keep going straight. Just before you reach the next intersection, you will see a bus stop on the right in front of a credit union.
Landmarks:
1. a bus stop
2. credit union

If you look around there should be a beige building on your right and a green awning. You want to head in the same direction as the the red building with a staircase and a green awning if you check your surrounding. Make a left turn at the intersection when you arrive. Follow the road until you reach another intersection. At this intersection make a left turn. You should be in an alley. If you go up a few steps there should be a bicycle leaning on a tree. There should be a white car next to the bike. Up ahead at least one step is a silver car and a light green car.
Landmarks:
1. beige building
2. green awning
3. a red building with a staircase and a green awning
4. a bicycle leaning on a tree
5. a white car next to the bike
6. a silver car
7. a light green car

Turn so your facing the intersection. You will take one step and be in the intersection. Turn Left, you will see some construction barriers on your left. Go one block and at the very next intersection go left again. Go about half a block or so and you will see another orange barricade on your left. There will be some tarps covering construction stuff and scaffolding. At the beginning of the barricade, there is an orange safety light.
Landmarks:
1. construction barriers
2. orange barricade
3. tarps covering construction stuff and scaffolding
4. orange safety light

{*navigation instructions*}
Landmarks:
<>

Figure 9: Prompt to extract landmarks from navigation instruction in Touchdown.

Map2seq:

Navigation Instructions (ID: 6197):

Head through the first intersection and at the next light make a right. Go past the next light and the Butcher Daughter will be on the far left corner. At the next light make a left and stop in front of Kings Avenue Tattooing.

Extracted Landmarks: "The Butcher Daughter", "Kings Avenue Tattooing"

Navigation Instructions (ID: 6205):

Head past the market and the cathedral and make a right at the light. At the next light with the Delicatessen on the corner make a left. Stop in front of the fire hall.

Extracted Landmarks: "a market", "a cathedral", "a Delicatessen", "a fire hall"

Navigation Instructions (ID: 6211):

Go to the end of the block and turn left. Pass More Parlour on the right and turn right at the lights. Go past the park on the left to the lights and turn left and take two steps. Stop at Straus Square on the right before the bike rental.

Extracted Landmarks: "More Parlour, "a park", "Straus Square", "a bike rental"

Touchdown:

Navigation Instructions (ID: 546):

You're going to go down the narrow street, not the big/main street here. Turn yourself so you've got that big mural of a guy with nunchucks at your back, and you're facing down the narrow street where you'll go in the same direction the parked cars are facing. Go down that street, and pass through the first intersection with the stop sign. At the second intersection, turn right. Go until you're nearly in the next intersection (right before you'd be standing on the crosswalk).

Extracted Landmarks: "mural of a guy with nunchucks", "parked cars", "stop sign", "crosswalk"

Navigation Instructions (ID: 580):

You're basically starting in an intersection. Move to the center of the intersection, and turn yourself so the restaurant with the bright yellow awnings and sidewalk barriers is on your right side (you'll pass it on your right as you walk down the street). Go down that street, with the yellow restaurant on your right, and go to the next intersection. Turn right. Look at the buildings on your right. A short way down the block you'll come to a bar with a wood bench out front. There is also a red velvet rope near the bench.

Extracted Landmarks: "restaurant with bright yellow awnings and sidewalk barriers", "bar with a wood bench", "red velvet rope"

Navigation Instructions (ID: 584):

Turn yourself around left so that you are going with the flow of traffic, there should be a green door on your right. Go forward and make a right turn at the first intersection. There will be a black awning on your right. Continue forward. When you come to the next intersection, make another right turn. As you get near the next intersection, you will see large red brick buildings on your right. You will see a pallet of green sandbags sitting along the sidewalk.

Extracted Landmarks: "green door black awning", "large red brick buildings", "pallet of green sandbags"

Figure 10: Landmarks extracted by GPT-3 using the 5-shot prompt for Map2seq and Touchdown.