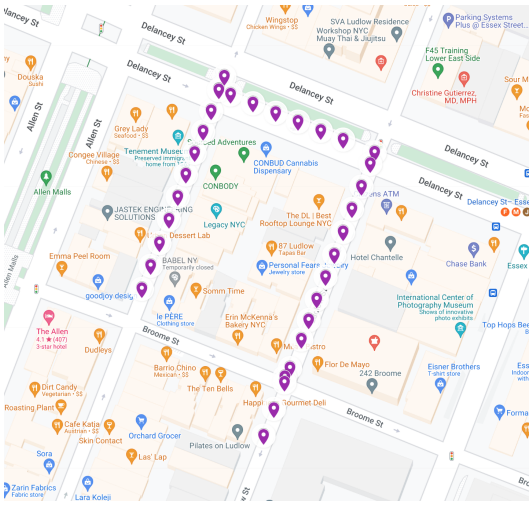
Follow the Beaten Path: The Role of Route Patterns on Vision-Language Navigation Agents Generalization Abilities

Kourosh T. Baghaei, Dieter Pfoser, Antonios Anastasopoulos
Department of Computer Science, George Mason University, USA
{kteimour, dpfoser, antonis}@gmu.edu

Abstract

Vision and language navigation (VLN) is a challenging task towards the creation of embodied agents that requires spatial and temporal reasoning over the instructions provided in natural language and aligning them with the visual perception of an environment. Although a number of methods and approaches have been developed, none achieves human level performance in outdoor settings (by up to 75 percent). The contributions of visual and language modalities to the success of VLN have been studied, however here we focus on an overlooked property of routes and show that navigational instructions can be represented as *patterns of actions* that also describe trajectory shapes. Through carefully crafted experiments, we show that agents generalization to unseen environments depends not only on visual and linguistic features, but also on the shape of trajectories presented to the model during the fine-tuning. Our experiments show that the diversity of patterns of actions during training is a key contributor to high success rates for agents. Last, we propose a solution based on data augmentation that fills the gap in missing patterns of training data. Our findings will guide researchers towards improved practices in the development and evaluation of VLN datasets and agents.[1]



Navigation Text of Route 641
You should be facing the correct direction when you load in. Begin by moving forward until you reach an intersection, and then take a right. Reorient yourself and take a right at the next intersection. Reorient yourself again and move forward though the next intersection. Stop three screens after this intersection. If you turn to the left slightly, there should be a traffic barrel near a shopping cart, which is in front of a red car.

The Pattern of Actions:
forward, right, forward, right, forward, stop.

Figure 1: Visualization of route 641 on Google Maps from TouchDown, along with navigation instructions and corresponding pattern of actions.

1 Introduction

Vision-language navigation (VLN) is a challenging research area that combines computer vision and natural language processing to enable embodied agents to navigate and understand their environment based on instructions provided in natural language. A typical solution to solving this problem is to train neural network architectures such as LSTM (Fried et al., 2018) and Transformers (Schumann and Riezler, 2022) from scratch. In contrast, using LLMs facilitates the development of modular agents (Shah et al., 2022; Schumann et al., 2024; Zhou et al., 2023) by taking advantage of reasoning capabilities learned through pre-training. Nonetheless, even with LLMs, a significant gap remains between human-level and agent-based performance when solving VLN tasks in outdoor settings (Schumann et al., 2024).

Eliminating such a performance gap requires a better understanding of the contributing factors to the success and failures of the agents. Zhu et al. (2022) studied token-level features of instructions and structural features of routes such as heading difference in turns. Schumann and Riezler (2022) focused on junction types for navigation.

[1] https://github.com/kbaghaei/beaten_path_vln

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 7986–8005

In this work, we focus on an overlooked property of navigational routes, which we call **Pattern of Actions** (PAct), which can be understood as the high-level shape of an agent trajectory. Figure 1 shows that each navigational pattern has a corresponding PAct. To the best our knowledge, PActs as a contributing factor to VLN agent performance have been overlooked.

We find that the agents' performance on out of sample test sets highly relies on the pattern of actions seen during the training, a phenomenon we call "pattern leakage". We show that removing some patterns of actions from training results in performance degradation in task completion by up to 8.9 percentage points. Based on our findings, we propose a data augmentation method to create synthetic training data that cover the missing patterns in the human generated dataset.

Our contributions can be summarized as follows:

1. We show that as an intrinsic feature of navigational trajectories, navigational patterns play an important role in model performance. This is reflected in success of the model in navigating routes with similar patterns, even with instructions that are from another routes. We introduce new data splits, where train and test data are separated to minimize pattern leakage, and show that agents largely fail on pattern generalization.
2. Successful VLN, beyond navigation, requires intermediate sub-tasks such as initial orientation, and stopping at the right point along the navigational path. We perform an in-depth analysis, comparing the fine-tuned agents on different data splits, showing that not observing patterns during training also deteriorates the agents' performance on these sub-tasks by up to 10 percentage points in F1 scores.
3. We explore a data augmentation method to create synthetic training data that makes up for the missing patterns in the human generated data, showing promising initial results.

2 Vision and Language Navigation

A navigation task can be defined as following instructions provided in natural language in order to ground a destination point within a given environment (Schumann et al., 2024). A navigation instruction $L = (w_1, w_2, ..., w_N)$ is a sequence of words in natural language that describes a navigation route $R = (n_1, n_2, ..., n_M)$. Each navigational route consists of multiple nodes in the navigational graph of the environment. Each node n_i in the navigational graph also has visual information v_i (a 360-degree panorama image). In each round of navigation, a VLN agent starts at an initial state s_1 and according to the instruction L and visual observation v_1 predicts a navigational action from the action space of {`FORWARD`, `LEFT`, `RIGHT`, `TURN_AROUND`, `STOP`}. After taking the action, it moves to another state, obtains another visual observation, and predicts a navigational action again. This loop continues until the agent decides the action `STOP`, or it runs out of action limit. A navigation is considered successful if the agent stops within one node distance of the destination point.

3 Patterns of Actions

Patterns of Actions (PAct) can be considered "principal components" of navigation trajectories. Consider the navigation instruction shown in Figure 1, and notice that it consists of largely two components: (a) directional information at key points where the agent should make turns, and (b) description of forward movements. The instructions also contain several references to landmarks.

We define PActs as the sequence of actions $a_1, a_2, ..., a_n$ of an agent, where any consecutive sequence of `forward` is represented as one single `forward` action. This way, this PAct stands for an abstract representation of the trajectory, capturing the ground truth actions at key decision points. For example, as depicted in Figure 1 (bottom), the navigational text can be summarized using the following PAct: `forward, right, forward, right, forward, stop`. Although moving forward might mean either one block or several kilometers, such a sequence of actions at key points can represent the structure of a navigational route. For brevity, we will represent each unique pattern with a hash, with the above example represented as `frfrfs`. Figure 1 also shows the actual route 641 on a map.

A PAct effectively also describes the shape of a trajectory. Therefore, throughout the paper we use the phrases *shape of trajectory* and *pattern of actions/PAct* interchangeably to emphasize the similarity of routes whose patterns of ground truth actions are equal.

Given our definition of PAct, Table 9 shows the number of unique patterns in our datasets. Compared to the number of samples, the number of unique PActs is 2 orders magnitude smaller, i.e. 66 unique patterns for 7352 samples in Map2Seq

(Schumann and Riezler, 2021) dataset, and 85 unique patterns for 9325 samples in TouchDown (Chen et al., 2018).

The datasets available in the literature share common PActs. In this work, we base our analysis on those PActs and perform experiments and ablation studies to show the contribution of **pattern leakage** in agents' performance. While bias of VLN models in prediction of actions has been studied before (Yu et al., 2020a), to the best of our knowledge, this is the first analysis of VLN approaches and datasets using such a pattern based approach.

4 Experimental Settings

4.1 LLM-based Agents

In our study, we utilize VELMA (Schumann et al., 2024). It is a state-of-the-art modular agent consisting of two main components: (i) the *Reasoning module* is an LLM that takes in instructions and textual description of visual observations and predicts a sequence of actions. We use LLaMA (Touvron et al., 2023) 7B, LLaMA 2, 7B, and Mistral 7B (Jiang et al., 2023) as the reasoning module of VLN agent. (ii) *Vision module*, which is a multimodal model for grounding landmarks referred in instructions to the visual observations. We use OpenCLIP (Cherti et al., 2022). Any landmark that is grounded by OpenCLIP is added to the prompt of the LLM as an observation. In our experiments, we ablate the visual information (OpenCLIP vs. No-Vision) both during fine-tuning and inference to report its effect. When we train a model not using visual information, we report it with suffix NV (e.g. Llama2-NV). Similar to Schumann et al. (2024), we fine-tune the models using LoRA (Hu et al., 2021) for 20 epochs and we choose the best model by shortest path distance (SPD) to the destination, on the development set. [2]

4.2 Datasets

We perform our experiments on two datasets: (i) TouchDown (TD; Chen et al., 2018), which consists of 9,326 navigational routes in Manhattan, NY, generated by human annotators through an ego-centric view similar to Google street view and (ii) Map2Seq (M2S; Schumann and Riezler, 2021), which consists of 7,672 routes in the same neighborhood as TouchDown. However, annotators provided navigational instructions by looking at the map of the route.

Seen and Unseen splits. The original train/dev/test splits of the TouchDown dataset contains routes covering the area of Manhattan. The train and test splits geographically overlap. However, a new split was proposed in (Schumann and Riezler, 2021) for both TouchDown and Map2Seq datasets so that the train and test samples are in *geographically separate* chunks. This split is called unseen. Throughout this paper, we refer to it as a baseline by *base-unseen*.

Dataset comparison. There are subtle differences in the construction of the datasets that are important for the following discussion:

- *Initial Direction:* in TouchDown, the follower agent is facing towards a random direction in the beginning of the navigation. As a result, the first piece of instruction describes how the follower agent should orient itself towards the correct direction. On the other hand, for Map2Seq, the agent is initially placed in the correct orientation towards the next move along the route. Note that both datasets are verified by other humans as followers to ensure that the instructions accurately describe the routes.
- *Route Structure* routes of Map2Seq are generated by finding the shortest path among two different points on the navigational graph. Given the grid-like map of Manhattan, this limits the number of patterns of actions for Map2Seq agents. However, TouchDown uses routes that are not necessarily shortest path and have arbitrary patterns.

Synthetic Data The main objective of data augmentation in our study is to create routes whose patterns are missing from the human generated dataset. We create 10160 navigational samples as follows. We first create all possible patterns with $n \leq 7$ number of turns, resulting in 508 and 127 unique patterns for TouchDown and Map2Seq respectively. We then fill-in template-based instructions that correspond to the patterns and add landmarks that are randomly sampled from the baseline training dataset. To add diversity to word choice and sentence structure, we also paraphrase the instructions using GPT4-o. With the final instructions and pattern of actions at hand, we can create the desired action sequence to end up with a complete training instance.

Note that since we use an LLM as the reasoning

[2] Based on the size of data splits, fine-tuning the models would take somewhere between 16 to 28 hours on an NVIDIA A100- 80GB GPU. Inference, would take 30 to 60 minutes on the same GPU.

backbone of the navigation agent, we only need to fine-tune the LLM on the textual training samples, simulating the vision component. Therefore, unlike the work of Wang et al. (2023b) and Yu et al. (2020a), the generated routes need not correspond to an actual route on a navigable graph of map. Moreover, due to the length of route segments of the street blocks in the actual graph of Manhattan, sampling routes with some patterns are either not possible or would result in too long routes. So, sampling routes from the actual navigational graph is not a feasible way of generating synthetic data. Additional details of our data augmentation method are in Appendix section D.

4.3 Evaluation Metrics

Interested in quantifying the effect of patterns in the training data on agent performance for 3 main tasks, we use the following metrics:

- *Task Completion (TC)* represents the percentage of successful navigation instances among all navigation instances in the test set (Schumann et al., 2024).
- *Overshoot Rate (OSR)* is the rate at which the agent reaches a destination but fails to stop at the destination.
- *Orientation* assesses how capable the model is in orienting the agent towards the correct direction in the beginning of the navigation. We use Precision, Recall and F1 scores.

5 Experiments and Results

We are interested in the generalization ability of agents with respect to the patterns presented to the model during training. To this end, we split the datasets into train and test sets based on patterns, fine-tune the models on these splits, and discuss the results. Both our datasets, TouchDown and Map2Seq, have only a limited number of unique patterns (PActs) of 85 and 63, respectively. Table 9 shows the number of samples in the train, dev(elopment), and test data using a base-unseen split. However, notice that train, test and dev datasets share patterns, which motivates our first experiment.

5.1 Swapping Instructions of Similar Paths

We noticed that patterns that are present in train data are also present in test data. This allows us to form the following hypothesis:

FT→Test	Swapped with	OpenCLIP	No-Vision
Same Train-Test Dataset			
TD→TD	base-unseen	20.9	11.48
	similar	4.97	2.82
	different	2.92	1.46
M2S→M2S	base-unseen	39.13	33.75
	similar	5.96	6.21
	different	1.88	1.38
Different Train-Test Datasets			
M2S→TD	base-unseen	6.17	5.31
	similar	2.96	2.89
	different	1.19	1.53
TD→M2S	base-unseen	23.5	22.75
	similar	4.56	5.32
	different	2.25	2.13

Table 1: FT: Fine-tune dataset. Task completion rate for base-unseen in 3 scenarios: Instructions swapped with similar PAct, different PAct, and base-unseen (no swapping). In *similar PActs* case, the agent succeeds by up to 5.96%, whereas its controlled scenario, *different PActs* , fails more often. Hence, showing the importance of PActs as a contributing factor to agents performance.

> If the PAct of a trajectory is a contributing factor, then swapping the instructions of one route with instructions of another route and still retaining its shape (PAct), then this should still result in a successful completion of the navigation task.

To test this hypothesis, we take a test set of the unseen data split and for each route in the test set, we randomly choose five other routes that have an identical PAct and use the instructions as substitute instructions. We omit the few routes that have fewer than five similar routes. For each route, we also randomly choose five instructions from routes with different PActs to aid in the validation of our hypothesis.

Table 1 shows the results of these experiments compared to the baseline (base-unseen). Across different experiments, the model completes the navigation task in up to 5% of the test cases even without any visual information. On the other hand, the task completion (TC) rate is lower for routes whose instructions are swapped with routes of different patterns. The TC rates for similar pattern replacements ("similar" rows in Table 1) are always higher than those for different patterns ("different" rows). Overall, the results support our hypothesis and emphasizes the importance of PActs in VLN.

5.2 Zero Pattern Overlap: Seen and Unseen Patterns

Our observations so far support the hypothesis that pattern leakage plays a role in dowsntream performance. To further study this phenomenon, we reverse the question. What if we train and test a model on carefully selected samples that will exhibit zero pattern leakage (i.e., no patterns are shared between the training and test data)?

We create a new data split in which no sample from the training data shares pattern with any of the samples in the test set, denoted as *Zero Pattern Overlap (0-pact)*. We group the data samples based on their patterns and sort them based on the number of samples within each group in descending order. We then assign the even-index samples to the training set odd-index samples to the test set, ensuring zero overlap. We also leave samples of one pattern for the development set. In the Appendix, Figure 2 illustrates this process. The resulting dataset has a 50-50 train-test split. Also, there is no common pattern among the train, development, and test sets. Note that, although we ensure no leakage within samples of each dataset, cross-dataset leakage (e.g. Map2Seq train to Touchdown test) is still possible.

To control for the effect of number of samples of data for training (compared to the base-unseen split where around 75% of the data is used for training, 10% for development and 15% for testing), we resample the base split –with leakage– so that the number of samples in the train, dev, test sets match that of 0-pact's. We label this split *base-p* and will use it as the fair baseline for comparison with 0-pact. The details of these splits are in Table 9.

Effect of Patterns Results. Table 2 shows that the model's performance drops noticeably (on TouchDown train-test), from 4.06% in Llama2 using vision, to 7.81% in no-vision scenario. The range of the performance drop is from 1.51% to 7.15% for other cases. This underlines the importance of seeing patterns during the training phase for the agent's ability to resolve test cases. We should also emphasize that the TC rates are also worse for no-vision cases in 0-pact split, i.e., in cases where the agent totally ignores visual observations during the inference or fine-tuning. For example, when the model is fine-tuned with no-vision, the performance drop from controlled to 0-pact ranges from 3.83% to 15.25%. This suggests that the model heavily relies on patterns to navigate.

FT → Test	Split	Llama2 OpenCLIP	Llama2 No-Vision	Llama2-NV
Same Train-Test Dataset				
TD → TD	base-p	**28.34**	**15.58**	**17.5**
	0-pact	24.28	7.77	2.25
M2S → M2S	base-p	**50.16**	**37.1**	**43.74**
	0-pact	43.01	34.7	39.72
Different Train-Test Datasets				
TD → M2S	base-p	**27.7**	**29.51**	**28.93**
	0-pact	22.24	25.32	16.27
M2S → TD	base-p	**7.31**	**5.08**	**6.51**
	0-pact	4.56	3.55	2.68

Table 2: Task Completion Rate (%) for Zero-Pattern-Overlap split. When train and test sets share PActs, the agent TC rate would be higher.

Visual Data Contamination. Given that 0-pact only separates the routes based on their patterns, the test samples can be from the same area the model has seen in the training data and potentially causing data contamination in 0-pact split. Nonetheless, even with this type of data contamination, there is an evident decrease in TC rate when the training and test samples do not share any patterns compared to the baselines (base-p).

The question that may be raised here is as to how a model fine-tuned with 0-pact split on M2S, and tested on M2S (38.13% with vision, 30% without vision) still performs comparable to that of the base-unseen scenario (39.12% with vision, 33.75% without vision), even though it has been trained on fewer (almost half) samples?

We hypothesize that this can be partly due to the *geographical overlap* in the 0-pact case. This question motivates our next experiment.

FT → Test	Split	Llama2-7b OpenCLIP	Llama2-7b No-Vision
Same Train-Test Dataset			
TD → TD	base-pg	**18**	**10.8**
	0-pact-geo	10.6	4.9
M2S → M2S	base-pg	**31.42**	**23.5**
	0-pact-geo	28	22.25
Different Train-Test Datasets			
TD → M2S	base-pg	**24.83**	**22.75**
	0-pact-geo	15.91	14.08
M2S → TD	base-pg	**5.6**	3.7
	0-pact-geo	5.4	**4**

Table 3: Task Completion Rate (%) for Zero Pattern and Geographical Overlap. As a general trend, the TC rate decreases when the PActs in test data are not familiar to the model (i.e. train and test sets do not share any PActs).

Image	Split	Llama2 Precision	Llama2 Recall	Llama2 F1	Llama2-NV Precision	Llama2-NV Recall	Llama2-NV F1
OpenCLIP	0-pact	**53**	43.06	42.09	-	-	-
	base-p	45.1	**55.82**	**44.79**	-	-	-
No-Vision	0-pact	**28.13**	27.27	13.33	13.26	14.5	8.48
	base-p	27.08	**33.58**	**24.07**	**30.27**	**35.16**	**28.9**

Table 4: Orientation results for Zero-Pattern-Overlap split. We use TouchDown as test and fine-tuning set. **Bolded** results are better performing between a zero-pattern-overlap case and its controlled split. Agents performance deteriorates when PActs in test set are not in train set.

Image	Split	Llama2 Precision	Llama2 Recall	Llama2 F1	Llama2-NV Precision	Llama2-NV Recall	Llama2-NV F1
OpenCLIP	0-pact-geo	42.22	46.15	48	-	-	-
	base-pg	**51.07**	**54.2**	**58.27**	-	-	-
No-Vision	0-pact-geo	23.4	32.84	17.25	23.4	32.84	17.25
	base-pg	**27.8**	**40.54**	**23.30**	**27.8**	**40.54**	**23.30**

Table 5: Orientation result for Zero Pattern and Geographical Overlap for TouchDown as test and fine-tuning set. **Bolded** results are better performing between a 0-pact-geo case and its controlled split. Generally, the agent's performance degrades when patterns in test set are not in train set. 0-pact-geo denotes the mean of 0-pact-geo-a and 0-pact-geo-b, while base-pg denotes mean of base-pg-a and base-pg-b.

5.3 Zero Pattern and Zero Geographical Overlap

To mitigate the influence of both geographical overlap and pattern overlap within the dataset, we further partition the data according to both geographic coordinates and patterns creating *Zero Patterns and Geographical Overlap* splits. Since the train and test set in base-unseen are geographically separate, if we take samples from its train set, whose patterns are different from samples in its test set, then we will have samples that have both geographical and pattern separation. So, similar to the 0-pact scenario, we group all data samples of base-unseen based on their patterns and sort them by the number of samples. Now, for each pattern, we can split the sampled into ones used for training and ones used for test. From the even indices, we take train samples, and from the odd indices we take test samples, to form a split with zero pattern and geographical overlap (denoted by *0-pact-geo-a*). We follow a reverse-sampling procedure (taking train samples from odd indices and test samples from even indices) to generate a second dataset from the remaining data *(0-pact-geo-b)*. To form the dev split, we randomly sample 15 and 20 percent of the test samples for TouchDown and Map2seq respectively. Appendix Figure 3 visualizes this process.

Since such a separation of data results in smaller datasets for train and test, we control for data size by creating two splits as baselines: *base-pg-a*, *base-pg-b*. We sample from base-unseen train to create train sets and sample from base-unseen test to create test sets, ensuring that the number of train-dev-test splits in base-pg-a and base-pg-b match to 0-pact-geo-a and 0-pact-geo-b respectively.

This way, the geographical separation of train and test splits in base-pg-a and base-pg-b are guaranteed, while they share patterns. The details of the data splits are listed in Table 9.

We fine-tune and test the models on these new splits of data. We report the average of pairs of 0-pact-geo-a and 0-pact-geo-b as 0-pact-geo, and average of base-pg-a and base-pg-b as base-pg. As a general trend in Table 3, for each pair of 0-pact-geo and base-pg the models performance deteriorates (from 5.9% to 7.4% where TouchDown was used for both training and testing). This reduction in model performance cannot be attributed to the size of training data as the performance on control cases (base-pg) is better. Furthermore, the potential data contamination that was present in 0-pact and base-p scenarios is not present here either. Hence, we can conclude that the patterns play a key role in the performance of the models.

5.4 Orientation

One key difference between the datasets of this study is that in TouchDown, the initial direction of the navigator agent is random whereas in Map2Seq the agent is facing towards the correct direction initially. This difference is also reflected in the instructions generated for each of the datasets. The first piece of instruction in TouchDown describes how the agent should orient itself towards the cor-

rect direction at the start of navigation. Therefore, an important sub-task in VLN is aligning towards the correct direction in the beginning of the navigation. In over 53% of test samples in TouchDown, the initial direction of the agent is incorrect, while that is the case for 0% for Map2Seq in both train and test splits.

The initial direction of the agent is encoded in the ground truth pattern of actions, represented by the first character. If the initial direction is towards the correct direction, then the ground truth pattern starts with a **f**orward as there is no need for the agent to make any turns. Otherwise, the agent might need to make a turn before moving forward, with the pattern starting with any of the `{l,r,t}` letters (which stand for `LEFT`, `RIGHT`, `TURN_AROUND` actions respectively).

We formulate the prediction of the initial action as a multi-class classification problem. To evaluate, we calculate F1 scores for each action and report macro-averaged Precision, Recall, and F1 scores.

Map2Seq neither teaches nor instructs the agent to make turns. When the test set is Map2Seq, the agent never makes any initial turns even when it is fine-tuned on TouchDown. Also, when the model is fine-tuned on Map2Seq, it rarely [3] makes any turns in the beginning since it has not learned to make any turns. Hence, for this analysis, we only focus on the Touchdown dataset.

The agent fails most often in orientation when the test dataset has patterns that are not present in the training data. Table 4 shows this general trend in the models' performance in the orientation sub-task. In the Zero Pattern Overlap scenario, the F1 score for orientation drops by 2.70% when the model is fine-tuned and tested on TouchDown using vision. Without vision data, the F1 score drops even more (by 10.74%) from 24.07% in controlled split to 13.33% in 0-pact.

Table 5 shows that the results of the Zero pattern and geographical overlap (0-pact-geo-x) scenario generally follow a similar trend. This indicates that the models are sensitive to the train-test separation of patterns for the orientation task as well.

5.5 Stopping

Accurately deciding where to stop is another crucial sub-task in vision and language navigation. Our error analysis on the base model showed that there is a significant number of what we term "overshoot errors". The agent reaches the destination, but erroneously continues moving instead of stopping. These are cases that could indeed have been successful had the agent stopped. We calculate the overshoot rate among all the cases that reached the destination as follows:

$$\text{Overshoot\_Rate} = \frac{\text{Overshoot}}{\text{Overshoot} + \text{Success}} \times 100,$$

Table 6 shows the results of overshoot rates in the Zero Pattern Overlap scenario. For the same train-test dataset scenarios, there is a consistent decrease in overshoot rates. However, in the scenario where the train and test datasets are different, overshoot rates do not always decrease from base-p to 0-pact split. This can be attributed to the fact that the cross dataset pattern leakage still exists.

Table 7 shows the overshoot rates[4] for the Zero Pattern and Geographical Overlap scenario. Generally, for each split and its controlled baseline split, the overshoot is lower in the baseline when fine-tuned on TouchDown dataset. However, when the model is fine-tuned on Map2Seq, the results are fairly similar. The overshoot rate is affected by the separation of patterns in one of two ways. One, it reduces the agents' generalization on routes with unseen patterns, leading to a reduction in task completion rate (TC). Two, in most of the overshoot scenarios, the agent is actually able to navigate the route and make it to the destination, but fails to stop at the right place. In such a case, the agent has actually followed a pattern similar to the ground truth pattern of the route. However, if a pattern is totally unfamiliar to the agent, the agent is less likely to reach the end of the route. Rather, it is more likely to make a wrong turn in the middle of the route. In turn, this would disqualify the route as an overshoot example. Hence, we conclude that the overlap in PActs, might affect the overshoot rate. The details of these scores are in Appendix Table 16.

5.6 The Effects of Data Augmentation

Table 8 compares fine-tuning our LLM solely with human generated data (baseline) with the case where the training data consists of human generated data plus the augmented data (Baseline + DA). In almost all cases, our simple data augmentation method increases task completion rate for both

[3] At most 2% in any of the test splits.

[4] We report the average of overshoot rates for 0-pact-geo-a and 0-pact-geo-b, as *0-pact-geo* and average of base-pg-a and base-pg-b as *base-pg*.

		Llama2		Llama2-NV
FT → Test	**Split**	**OpenCLIP**	**No-Vision**	
Same Train-Test Dataset				
TD → TD	0-pact	54.85	60.92	77.67
	base-p	**46.4**	**56.61**	**45.22**
M2S → M2S	0-pact	35.62	47.63	36.81
	base-p	**26.81**	**47.39**	**30.04**
Different Train-Test Dataset				
TD → M2S	0-pact	**17.45**	**29.08**	46.74
	base-p	19.68	29.97	**27**
M2S → TD	0-pact	76.08	**79.89**	84.72
	base-p	**75.98**	82.82	**77.59**

Table 6: Overshoot Rate for Zero Pattern Overlap split. In general, when there is no common PActs between train and test sets, the overshoot rate is higher.

		Llama2		
FT → Test	**Split**	**OpenCLIP**	**No-Vision**	**Llama2-NV**
Same Train-Test Dataset				
TD → TD	0-pact-geo	79.69	84.03	84.03
	base-pg	**58.24**	**66.17**	**66.17**
M2S → M2S	0-pact-geo	**37.44**	**51.84**	**51.84**
	base-pg	38.55	51.95	51.95
Different Train-Test Dataset				
TD → M2S	0-pact-geo	56.89	63.12	63.12
	base-pg	**27.10**	**39.41**	**39.41**
M2S → TD	0-pact-geo	**77.63**	**84.11**	**84.11**
	base-pg	80.72	84.21	84.21

Table 7: Overshoot Rate for Zero Pattern and Geographical Overlap (0-pact-geo). The overshoot rate is significantly higher when the train-test sets do not share PActs (0-pact-geo) compared to baselines (base-pg) for the TD dataset. Results are fairly similar to each other when fine-tuning on M2S. We conclude pattern overlap *might* affect overshoot rate.

		Llama2	
FT → Test	**Image**	**Baseline**	**Baseline + DA**
Same Train-Test Dataset			
TD → TD	No-Vision	**13.8**	13.13
	OpenCLIP	**23.22**	22.56
M2S → M2S	No-Vision	26.5	**31.5**
	OpenCLIP	36.75	**39.25**
Different Train-Test Datasets			
TD → M2S	No-Vision	25.5	**26.1**
	OpenCLIP	23	**23.5**
M2S → TD	No-Vision	3.58	**3.8**
	OpenCLIP	4.98	**5.1**

Table 8: Task Completion Rate (%) for Baseline vs. Data Augmentation for Llama 2. Simple data augmentation can potentially enhance performance overall.

models, in M2S by almost 2.5 percentage points. When both train and test are from TouchDown, it performs slightly worse, which we think can be attributed to the fact that the agent's initial direction is random in TouchDown and the vision component does not align the direction of the landmark at the starting point. We discuss the data augmentation results in more detail in Appendix Section D.

6 Related Work

Navigation of environments by visual perception and language instruction is a well studied problem for both indoor (Anderson et al., 2017; Ku et al., 2020) and outdoor settings (Chen et al., 2018; Schumann and Riezler, 2021; Vasudevan et al., 2021a). Model-wise, various methods have been proposed to perform this task using LSTMs (Fried et al., 2018), transformer based models (Schumann and Riezler, 2022), and LLM-based agents (Schumann et al., 2024). From a dataset perspective, navigational routes do not present enough diversity to models as they are sampled using shortest path algorithms (Yu et al., 2020a). This presents biases in action space that leads to models failing to generalize to unseen routes (Yu et al., 2020a). We study such a bias through the lens of sequence of actions that a LLM predicts and how the patterns of sequences of actions affect the models' performance. Due to high cost of human annotation, Fried et al. (2018), Yu et al. (2020a), and Wang et al. (2023b) propose methods to create synthetic data to add more diversity in the distribution of samples in the training data. A more in-depth discussion of the related work is available in Appendix 6.

7 Conclusion

Our evaluation of LLM-based vision and language navigation agents shows that navigation instructions contain an abstract representation of the shape of a trajectory, which captures the pattern of actions an agent must take to perform the navigation task. Using this patterns as the basis of our evaluation, we show that VLN agents' are less likely to generalize to routes whose patterns are not present in training data. We find that using diverse patterns during the training phase improves the agents' performance. We additionally propose a simple data augmentation method that has the potential to improve the agents' task completion rate.

A more comprehensive study of data augmentation is required in future work. Also, the effect of PActs on indoor environments should be analyzed, as our study focuses solely on outdoor settings. Another track of study is whether the pattern of actions in instructions, affect performance of VLN models in continuous environments. Our suggestion for the development of new datasets for VLN is to generate navigational routes with a higher diversity of patterns of actions to improve performance, and to consider this variable when evaluating VLN agents

and their generalization capabilities.

Limitations

The limitations of our study can be summarized as follows:

VLN Agents. We do not discuss the effect of patterns on VLN agents that are LSTM (Fried et al., 2018) or Transformer-based (Schumann and Riezler, 2022) that use end-to-end training. This is because:

1. Transformer-based models are superior in performance compared to LSTM based models on VLN tasks. (Schumann and Riezler, 2022)
2. LLMs are pre-trained on huge and diverse datasets and we can take advantage of such models by fine-tuning them.
3. Although we propose a data augmentation method, we do not explore and discuss every factor of the dataset augmentation method. Rather, we present an experiment using a fixed set of hyper-parameters, such as dataset size, various paraphrasing methods, different types of landmark sampling that worked best in our experiments. Nonetheless, we show the potential of our approach and leave any further analysis of our method for future exploration.

Simplification Assumptions. The agent of our study navigates in a discrete environment. The actions of the agent are considered complete. However, the effect of PActs in a continuous setting is an open research question.

Diversity of Languages. We only consider the English language and leave the study of PActs and VLN in other languages to future work.

Granularity of Contributing Factors. We do not consider token-wise analysis as it is has been studied in the literature (Zhu et al., 2022). Also, we do not consider fine-grained structural features such as junction types and directional changes since they have been thoroughly analyzed and discussed by Schumann and Riezler (2022). Rather, we focus on the route structure, which is overlooked in the literature.

Ethics Statement

In this study, we use panorama images of street view published by Google (Mirowski et al., 2018). Privacy and ethics concerned with the dataset have been addressed by blurring individuals' faces in the image data. Since we conducted our experiments in a simulated environment, there is no risk of damage or injury. However, deploying and experimenting VLN in real world environments would require additional, extensive safety measurements which are beyond the scope of this study.

Acknowledgments

We are thankful to the reviewers and meta-reviewers for their constructive feedback. This project is supported by the National Science Foundation under grant III-2127901.

References

Mohamed Aghzal, Erion Plaku, and Ziyu Yao. 2023. Can large language models be good path planners? a benchmark and investigation on spatial-temporal reasoning. *ArXiv*, abs/2310.03249.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. 2017. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

Howard Chen, Alane Suhr, Dipendra Kumar Misra, Noah Snavely, and Yoav Artzi. 2018. Touchdown: Natural language navigation and spatial reasoning in visual street environments. *CoRR*, abs/1811.12354.

Peihao Chen, Xinyu Sun, Hongyan Zhi, Runhao Zeng, Thomas H. Li, Gaowen Liu, Mingkui Tan, and Chuang Gan. 2023. A2nav: Action-aware zero-shot robot navigation by exploiting vision-and-language ability of foundation models. *ArXiv*, abs/2308.07997.

Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. 2022. Reproducible scaling laws for contrastive language-image learning.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. *ArXiv*, abs/1806.02724.

Karl Moritz Hermann, Mateusz Malinowski, Piotr Mirowski, Andras Banki-Horvath, Keith Anderson, and Raia Hadsell. 2020. Learning to follow directions in street view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 07, pages 11773–11781.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and

Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. 2019. General evaluation for instruction conditioned navigation using dynamic time warping. *ArXiv*, abs/1907.05446.

Vihan Jain, Gabriel Ilharco, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. 2019. Stay on the path: Instruction fidelity in vision-and-language navigation. In *Annual Meeting of the Association for Computational Linguistics*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. 2020. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Computer Vision – ECCV 2020*, pages 104–120, Cham. Springer International Publishing.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. 2020. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. *CoRR*, abs/2010.07954.

Shuhei Kurita and Kyunghyun Cho. 2020. Generative language-grounded policy in vision-and-language navigation with bayes' rule. *ArXiv*, abs/2009.07783.

Jialu Li, Aishwarya Padmakumar, Gaurav Sukhatme, and Mohit Bansal. 2024. Vln-video: Utilizing driving videos for outdoor vision-and-language navigation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):18517–18526.

R. Liu, X. Wang, W. Wang, and Y. Yang. 2023. Bird's-eye-view scene graph for vision-language navigation. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10934–10946, Los Alamitos, CA, USA. IEEE Computer Society.

Piotr Mirowski, Matthew Koichi Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, Koray Kavukcuoglu, Andrew Zisserman, and Raia Hadsell. 2018. Learning to navigate in cities without a map. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 2424–2435, Red Hook, NY, USA. Curran Associates Inc.

Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. 2020. 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.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Raphael Schumann and Stefan Riezler. 2021. 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*.

Raphael Schumann and Stefan Riezler. 2022. 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. Association for Computational Linguistics.

Raphael Schumann, Wanrong Zhu, Weixi Feng, Tsu-Jui Fu, Stefan Riezler, and William Yang Wang. 2024. Velma: Verbalization embodiment of llm agents for vision and language navigation in street view.

Dhruv Shah, Blazej Osinski, Brian Ichter, and Sergey Levine. 2022. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on Robot Learning*.

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. 2023. Llama: Open and efficient foundation language models.

Arun Balajee Vasudevan, Dengxin Dai, and Luc Van Gool. 2021a. Talk2nav: Long-range vision-and-language navigation with dual attention and spatial memory. *International Journal of Computer Vision*, 129(1):246–266.

Arun Balajee Vasudevan, Dengxin Dai, and Luc Van Gool. 2021b. Talk2nav: Long-range vision-and-language navigation with dual attention and spatial memory. *International Journal of Computer Vision*, 129:246–266.

Hanqing Wang, Wei Liang, Luc Van Gool, and Wenguan Wang. 2023a. Dreamwalker: Mental planning for continuous vision-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10873–10883.

Z. Wang, J. Li, Y. Hong, Y. Wang, Q. Wu, M. Bansal, S. Gould, H. Tan, and Y. Qiao. 2023b. Scaling data generation in vision-and-language navigation. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11975–11986, Los Alamitos, CA, USA. IEEE Computer Society.

Z. Yang, A. Majumdar, and S. Lee. 2023. Behavioral analysis of vision-and-language navigation agents. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, Los Alamitos, CA, USA. IEEE Computer Society.

Felix Yu, Zhiwei Deng, Karthik Narasimhan, and Olga Russakovsky. 2020a. Take the scenic route: Improving generalization in vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. 2020b. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2633–2642.

Yusheng Zhao, Jinyu Chen, Chen Gao, Wenguan Wang, Lirong Yang, Haibing Ren, Huaxia Xia, and Si Liu. 2022. Target-driven structured transformer planner for vision-language navigation. *Proceedings of the 30th ACM International Conference on Multimedia*.

Gengze Zhou, Yicong Hong, and Qi Wu. 2023. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. In *AAAI Conference on Artificial Intelligence*.

Wanrong Zhu, Yuankai Qi, Pradyumna Narayana, Kazoo Sone, Sugato Basu, Xin Wang, Qi Wu, Miguel Eckstein, and William Yang Wang. 2022. Diagnosing vision-and-language navigation: What really matters. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5981–5993, Seattle, United States. Association for Computational Linguistics.

A Extended Related Work

Vision and Language Navigation Following navigational instructions to reach destination in a navigable environment is a well studied topic. Various datasets and benchmarks have been proposed for indoor navigation such as R2R (Anderson et al., 2017), RxR (Ku et al., 2020), Qi et al. (2020) and Krantz et al. (2020), typically providing a navigational graph along with panorama images of each point on the graph. (Krantz et al., 2020) expand the indoor navigation to a continuous environment and (Wang et al., 2023a) perform navigation in continuous environment by using an internal world model.

For outdoor navigation, several datasets and models have been proposed: StreetLearn (Mirowski et al., 2018) presents a dataset of panorama views of Google Street View, TouchDown (Chen et al., 2018) offer a dataset of over 9300 random samples of navigational trajectories of Manhattan annotated by humans through ego-centric view, and Map2Seq (Schumann and Riezler, 2021) release over 7000 samples of random navigational trajectories within the same area as TouchDown (note however that the instructions are created through a bird's eye view). StreetNav (Hermann et al., 2020) propose an RL framework for VLN and sample 613000 routes with corresponding instructions from Google API, and Talk2Nav (Vasudevan et al., 2021b) create a dataset of 10700 navigational routes with corresponding annotations that follow the structure of description of landmark and directional instructions. (Li et al., 2024) use BDD100k (Yu et al., 2020b) dataset which consists of driving videos as training data to enhance VLN performance in outdoor settings.

While VLN was previously performed using mostly LSTM based models (Fried et al., 2018; Hermann et al., 2020), transformer-based models that are trained end-to-end have been proposed as well (Schumann and Riezler, 2022). All aforementioned studies explore discriminative methods in performing VLN tasks in which the agents predict next action based on the history of actions, trajectory and instructions. In contrast, (Kurita and Cho, 2020) propose a generative language-grounded policy in which a language model predicts all possible instructions given the past states and actions. Then, the agent takes actions whose probability of generating instructions are maximum.

LLMs and Modular Agents The promising reasoning ability of large language models on linguistic task has attracted researchers interest in path planning (Aghzal et al., 2023). Also, it has enabled the development of modular agents such as LM-Nav (Shah et al., 2022), NavGPT (Zhou et al., 2023), A2Nav(Chen et al., 2023), and VELMA (Schumann et al., 2024). In these agents, the task of VLN is performed by having an LLM perform as the reasoning and planner component and having other multi-modal models such as CLIP (Radford et al., 2021) as a visual alignment module.

Topology and Route Structure Rather than solely relying on the history of past visual observations and taken actions, representing the topology of the navigable environment as an abstract graph has been studied in various studies (Zhao et al., 2022; Liu et al., 2023). Addition of such a mental map of the environment, enhances the performance

of VLN agents. However, these studies do not discuss the effect of topology and patterns of routes on agents performance. Yu et al. (2020a) analyze VLN agents through the lens of predicted actions. They show that VLN agents are more likely to predict certain actions that have been more present in the training data. However, their study is different from ours in two ways. First, their study is limited to indoor environments where the length and complexity of routes are lower compared to the outdoor settings. Second, they analyze the probability of actions regardless of their past sequence of actions. In our paper however, we analyze the predicted actions as a whole sequence and discuss whether the sequences of actions present in the training set, affect the performance of the agents on the test set.

Model Behaviour Analysis Evaluation of deep generative models is both important and challenging. For VLN, various evaluation methods have been proposed. While methods have been proposed for assessing similarity of trajectories (Ilharco et al., 2019; Jain et al., 2019), these scores do not reveal any further details on how the models perform. For outdoor VLN, Schumann and Riezler (2022) perform various ablation experiments and show that structural features of routes such as junction type and difference in heading have higher weight on the performance of models compared to visual cues. Also, Zhu et al. (2022) show that for indoor, the models use object tokens and directional tokens for navigation. Whereas, for the outdoor, the models' performance mostly depends on directional tokens. Yang et al. (2023) propose a method for intervening with the instructions given to the agent and evaluating its sensitivity to the interventions. In this way, they analyze skill-specific capabilities of VLNs. Our study differs from the previous ones in several ways: First, unlike these studies, we focus on LLM-based models. As the LLMs provide strong reasoning capabilities that can be incorporated in navigational tasks with fine-tuning. Hence, eliminating the need to train a model from scratch. Second, we do not perform a token-wise analysis. Rather, we focus on the structure of navigational routes. Nonetheless readers can refer to (Zhu et al., 2022) for a holistic analysis on token level evaluation of VLNs. Finally, we focus on the outdoor navigation only as it is understudied.

Data Augmentation Due to the high cost of collection of human generated data, various studies propose methods of data augmentation. An LSTM-based model, pretrained on the human-generated data can be used for sampling instructions that correspond to randomly sampled routes (Fried et al., 2018; Yu et al., 2020a). ScaleVLN (Wang et al., 2023b) scales the data augmentation by creating navigable graphs and sampling routes from newly created graphs. However, in generating VLN instructions, they take a similar approach to Fried et al. (2018) by employing an LSTM. Our study differs from these studies in two ways. First, they focus on indoor, whereas our study focuses on outdoor environments where the navigational routes are longer and the landmarks and objects are more diverse. Second, since we use an LLM for geospatial reasoning, we do not need to rely on an actual navigable graph to sample navigational routes. Rather, the sequence of actions along with the instructions corresponding to them would suffice for training the model.

				Train		Dev		Test		
Split name	**GS**	**PS**	**Dataset**	**#S**	**#P**	**#S**	**#P**	**#S**	**#P**	**PO**
base-unseen	✓		TD	6,770	74	800	50	1,507	66	58
			M2S	5,737	37	800	31	800	31	28
0-pact		✓	TD	4783	42	286	40	4256	40	0
			M2S	3889	21	306	19	3477	19	0
base-p			TD	4781	73	286	63	4258	72	63
			M2S	3899	36	306	35	3467	35	31
0-pact-geo-a	✓	✓	TD	3506	37	160	35	994	35	0
			M2S	2892	20	150	17	633	17	0
base-pg-a	✓		TD	3506	74	160	74	994	69	69
			M2S	2892	37	150	36	633	36	31
0-pact-geo-b	✓	✓	TD	3264	37	160	34	993	34	0
			M2S	2892	17	150	20	633	20	0
base-pg-b	✓		TD	3264	74	160	69	993	69	69
			M2S	2892	37	150	36	633	36	36

Table 9: The number of samples and the number of PActs in each train, dev, and test set for different data splits. **GS:** Geographical Separation. **PS:** PActs Separation. **PO:** PActs Overlap, the number of common PActs between train and test sets. **#S:** Number of Samples. **#P:** Number of PActs.

B Data Separation

Figures 2 and 3 visually show the process of creating data splits of *Zero Pattern Overlap* and *Zero Pattern and Geographical Overlap* respectively.

C Additional Results

Here we show our complete results on Llama1-hf 7B, Llama2-hf 7B and Mistral 7B v0.1.

Test Dataset	Finetune Dataset	Scenario	Llama1-7b		Llama2-7b		Mistral-7b-v0.1	
			OpenCLIP	No-Vision	OpeCLIP	No-Vision	OpenCLIP	No-Vision
TouchDown	TouchDown	base-unseen	20.9	11.48	23.22	13.8	10.42	7.03
	Map2Seq	base-unseen	6.17	5.31	4.98	3.58	8.69	6.9
Map2Seq	TouchDown	base-unseen	23.5	22.75	23	25.5	5.62	6
	Map2Seq	base-unseen	39.12	33.75	36.75	26.5	35	32.62
TouchDown	TouchDown	base-p	**30.48**	**15.1**	**28.34**	**15.58**	**14.92**	**8.06**
		0-pact	5.82	3.19	24.28	7.77	7.05	3.05
	Map2Seq	base-p	**7.64**	**5.49**	**7.31**	**5.08**	**5.83**	**2.96**
		0-pact	2.53	1.87	4.56	3.55	2.02	1.36
Map2Seq	TouchDown	base-p	**30.95**	**26.62**	**27.7**	**29.51**	**8.57**	**9.49**
		0-pact	16.28	15.05	22.24	25.32	3.52	2.74
	Map2Seq	base-p	**49.52**	**38.94**	**50.16**	**37.1**	**39.57**	**25.65**
		0-pact	38.13	30	43.01	34.7	35.13	21.81
TouchDown	TouchDown	base-pg-a	**18.1**	**11.1**	**18**	**11.2**	**16.3**	**9.3**
		0-pact-geo-a	6.1	3.8	13.6	6.1	10.2	3.5
		base-pg-b	**20**	**11.2**	**18**	**10.4**	**10.2**	**6.5**
		0-pact-geo-b	3.2	1.8	7.6	3.7	1.6	0.9
	Map2Seq	base-pg-a	**7**	3.5	4.1	3.7	**3.2**	**2.3**
		0-pact-geo-a	5.5	**4.2**	**4.9**	**3.9**	1.7	1.1
		base-pg-b	**5.4**	3.9	**6.7**	**4.3**	**5.2**	**3.1**
		0-pact-geo-b	4.2	**4**	6.3	3.5	1	0.5
Map2Seq	TouchDown	base-pg-a	**17.5**	**18.33**	**26.17**	**24**	**19.83**	**21.17**
		0-pact-geo-a	15.83	13.33	16.83	13.67	14.83	13.5
		base-pg-b	**20.83**	**21.66**	**23.5**	**21.5**	**4.67**	**6.17**
		0-pact-geo-b	7.66	6.5	15	14.5	4.17	2.67
	Map2Seq	base-pg-a	**34.83**	**28.66**	25.17	19.83	**17.17**	**16.83**
		0-pact-geo-a	28.49	22.5	**31.67**	**26**	7.83	8.67
		base-pg-b	**37.33**	**25.83**	**37.67**	**27.17**	**34.67**	**26.83**
		0-pact-geo-b	20.83	18.66	24.33	18.5	13.83	11.17

Table 10: Task Completion (TC) rate for Llama1 and Mistral, fine-tuned using vision. In each pair of 0-pact (0-pact-geo-x), and its controlled baseline, base-p (base-pg-x), the TC rate is higher in baseline. This indicates the contribution of PAct in agent's generalization to out of sample test sets.

Test Dataset	Finetune Dataset	Split	Llama1-7B-NV	Llama2-7B-NV	Mistral-7B-NV
TouchDown	TouchDown	base-unseen	14	14.4	10.95
	Map2Seq	base-unseen	6.64	3.45	2.65
Map2Seq	TouchDown	base-unseen	19.62	21.62	23.88
	Map2Seq	base-unseen	33.62	27.75	26.25
TouchDown	TouchDown	base-p	**17.22**	**17.5**	**11.14**
		0-pact	6.08	2.25	2.47
	Map2Seq	base-p	**7.07**	**6.51**	**6.48**
		0-pact	1.9	2.68	1.86
Map2Seq	TouchDown	base-p	**27.67**	**28.93**	11.13
		0-pact	19.47	16.27	**22.61**
	Map2Seq	base-p	**42.8**	**43.74**	**34.45**
		0-pact	33.86	39.72	18.34
TouchDown	TouchDown	base-pg-a	**11.1**	**11.2**	**9.3**
		0-pact-geo-a	3.8	6.1	3.5
		base-pg-b	**11.2**	**10.4**	**6.5**
		0-pact-geo-b	1.8	3.7	0.9
	Map2Seq	base-pg-a	3.5	3.7	**2.3**
		0-pact-geo-a	**4.2**	**3.9**	1.1
		base-pg-b	3.9	**4.3**	**3.1**
		0-pact-geo-b	**4**	3.5	0.5
Map2Seq	TouchDown	base-pg-a	**18.33**	**24**	**21.17**
		0-pact-geo-a	13.33	13.67	13.5
		base-pg-b	**21.67**	**21.5**	**6.17**
		0-pact-geo-b	6.5	14.5	2.67
	Map2Seq	base-pg-a	**28.67**	19.83	**16.83**
		0-pact-geo-a	22.5	**26**	8.67
		base-pg-b	**25.83**	**27.17**	**26.83**
		0-pact-geo-b	18.67	18.5	11.17

Table 11: Task Completion (TC) rate for fine-tuned models without using vision. Between each split and its controlled baseline, the best performing score is **bolded**. As a general trend, the TC rate drops when train and tests do not share PActs, even when no visual information was used in training. This renders the PActs as a contributing factor in agents performance.

	Llama1-7B-NV			Llama2-7B-NV			Mistral-7B-NV		
Split	**Precision**	**Recal**	**F1**	**Precision**	**Recal**	**F1**	**Precision**	**Recal**	**F1**
base-unseen	24.81	28.41	23.44	30.42	42.45	31.16	51.49	25	17
Zero Pattern Overlap									
0-pact	16.31	24.83	8.74	13.26	14.5	8.48	10.35	12.93	6.88
base-p	**31.93**	**42.02**	**32.66**	**30.27**	**35.16**	**28.9**	**39.78**	**37.82**	**38.68**
Zero Pattern and Geographical Overlap									
0-pact-geo-a	19.87	**30.61**	**19.38**	17.1	29.65	17.25	23.13	32.75	15.43
base-pg-a	**20.78**	25.02	13.9	**24.26**	**34.59**	**22.47**	**26.06**	**35.15**	**20.77**
0-pact-geo-a	**36.54**	37.24	24.32	29.7	36.04	**25.33**	**26.89**	**38.25**	**25.2**
base-pg-b	29.92	**50.58**	**28.28**	**31.34**	**46.49**	24.14	22.47	34.44	20.76

Table 12: Orientation : Models fine-tuned and evaluated without using vision on TouchDown dataset. Since the orientation of agent in the beginning of navigation in Map2Seq dataset is towards the route, and there is no instructions for orientation, when a model is fine-tuned on Map2Seq, it does not learn to make any turns in the start of navigation. Therefore, this table only shows results of experiments on TouchDown dataset only. The orientation task requires the agent to make turns based on its surrounding visual cues. Even though, no visual information is used by the agents in these experiments, when train and test sets do not share any patterns, the performance of the agents degrade.

				Llama1-7B			Llama2-7B			Mistral-7B		
Test Dataset	**Fine-Tune Dataset**	**Image**	**Split**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
TouchDown	TouchDown	CLIP	base-unseen	51.4	53.65	52.07	48.92	53.7	50.74	49.88	50.97	50.2
		No-Vision	base-unseen	26.37	36.64	25.99	29.94	43.33	31.29	30.64	36.39	29.15
	Map2Seq	CLIP	base-unseen	23.09	24.93	12.62	20.09	36.26	19.84	22.63	32.03	18.76
		No-Vision	base-unseen	23.09	24.93	12.62	21.28	36.26	19.56	23.22	30.67	18.43
TouchDown	TouchDown	CLIP	0-pact-geo-a	53.36	40.74	39.99	46.12	53.29	48	46.41	47.85	**46.94**
			base-pg-a	58.94	45.67	**44.68**	47.52	54.2	49.2	43.86	47.04	44.81
			0-pact-geo-b	76.77	39.15	38.55	38.32	39.01	38.5	53.01	43.42	40.01
			base-pg-b	57.04	54.88	**54.56**	54.63	62.34	**55.41**	62.36	60.62	**61.28**
		No-Vision	0-pact-geo-a	19.87	30.61	**19.38**	17.1	29.65	17.25	23.13	32.75	15.43
			base-pg-a	20.78	25.02	13.9	24.26	34.59	**22.47**	26.06	35.15	**20.77**
			0-pact-geo-b	36.54	37.24	24.32	29.7	36.04	**25.33**	26.89	38.25	**25.2**
			base-pg-b	29.92	50.58	**28.28**	31.34	46.49	24.14	22.47	34.44	20.76
	Map2Seq	CLIP	0-pact-geo-a	11.71	26.84	12.97	13.24	26.77	**13.48**	14.7	34.74	16.53
			base-pg-a	17.55	28.96	**16.74**	15.87	24.95	12.89	26.26	33.23	**21.35**
			0-pact-geo-b	17.86	28.28	17.12	25.31	35.44	**22.3**	17.8	24.95	**13.92**
			base-pg-b	26.29	31.9	**19.93**	22.54	31.9	19.5	11.66	24.79	12.69
		No-Vision	0-pact-geo-a	11.74	26.71	12.95	14.3	26.9	**13.73**	14.29	34.74	16.2
			base-pg-a	18.18	28.96	**16.89**	15.87	24.95	12.89	20.28	29.06	**17.36**
			0-pact-geo-b	21.09	32.13	**20**	21.13	31.64	**19.33**	26.68	24.95	**13.91**
			base-pg-b	39.01	29.78	18.07	17.96	26.9	15.37	15.55	24.84	12.7
TouchDown	TouchDown	CLIP	0-pact	64.75	46.15	**47.59**	53	43.06	42.09	32.82	39.16	31.61
			base-p	45.14	57.04	44.94	45.1	55.82	**44.79**	59.58	52.85	**55.1**
		No-Vision	0-pact	29.43	28.79	20.35	28.13	27.27	13.33	15.93	32.4	14.33
			base-p	32.3	39.69	**27.43**	27.08	33.58	**24.07**	31.85	37.31	**25.38**
	Map2Seq	CLIP	0-pact	18.64	28.3	12.74	16.06	28.6	12.74	14.98	25.54	9.34
			base-p	16.78	26.33	**14.74**	22.36	27.28	**15.96**	23.27	29.77	**18.5**
		No-Vision	0-pact	19.17	28.01	12.46	16.96	31.13	14.07	9.32	25.25	8.88
			base-p	15.92	25.85	**14.21**	20.22	27.25	**15.89**	20.94	28.31	**16.99**

Table 13: Precision, Recall and F1 scores for Orientation task. Between each pair of data split and its corresponding baseline, the best performing F1 score is **bolded**.

			Llama1-7B		Llama2-7B		Mistral-7B	
Test Dataset	**Fine-Tune Dataset**	**Split**	**OpenCLIP**	**No-Vision**	**OpenCLIP**	**No-Vision**	**OpenCLIP**	**No-Vision**
TouchDown	TouchDown	base-unseen	**46.88**	**60.59**	**42.53**	**54.98**	**69.22**	**73.03**
	Map2Seq	base-unseen	77.26	80.68	82.64	86.92	71.08	76.94
Map2Seq	TouchDown	base-unseen	**14.55**	**25.1**	**22.03**	**23.88**	47.67	42.86
	Map2Seq	base-unseen	24.94	38.78	37.31	57.26	**13.58**	**29.46**
TouchDown	TouchDown	0-pact-geo-a	88.18	87.66	75.05	77.32	81.65	87.59
		base-pg-a	**55.09**	**67.54**	**56.94**	**64.44**	**63.94**	**72.07**
		0-pact-geo-b	93.19	95.6	84.33	90.75	96.38	97.18
		base-pg-b	**54.13**	**66.77**	**59.55**	**67.9**	**69.28**	**74.21**
	Map2Seq	0-pact-geo-a	**70.27**	**78.12**	**73.37**	**79.03**	92.51	94.91
		base-pg-a	76.51	87.41	84.23	84.9	**85.39**	**89.59**
		0-pact-geo-b	87.9	87.95	81.9	89.2	97.21	98.57
		base-pg-b	**80.78**	**85.17**	**77.21**	**83.52**	**84.71**	**90.37**
Map2Seq	TouchDown	0-pact-geo-a	53.88	60.2	47.12	56.15	66.29	68.24
		base-pg-a	**28.57**	**45.27**	**22.66**	**33.02**	**36.02**	**45.02**
		0-pact-geo-b	79.19	83.4	66.67	70.1	87.37	92.42
		base-pg-b	**23.78**	**34.34**	**31.55**	**45.8**	**36.36**	**28.85**
	Map2Seq	0-pact-geo-a	**25.97**	**47.47**	**28.57**	**42.22**	72.67	73.2
		base-pg-a	32.14	48.35	42.8	51.43	**55.22**	**61.3**
		0-pact-geo-b	55.36	62.03	46.32	61.46	72.7	78.25
		base-pg-b	**33.13**	**54.55**	**34.3**	**52.48**	**33.76**	**50**

Table 14: Overshoot Rate (OSR) among different models and data splits. For each pair of Zero PActs and Geographical Overlap (0-pact-geo-x) and control splits (base-pg-x), the best performing split is **bolded**.

Test Dataset	Fine-Tune Dataset	Split	Llama1-7B-NV	Llama2-7B-NV	Mistral-7B-NV
TouchDown	TouchDown	base-unseen	46.88	42.53	69.22
	Map2Seq	base-unseen	77.26	82.64	71.08
Map2Seq	TouchDown	base-unseen	**14.55**	**22.03**	47.67
	Map2Seq	base-unseen	24.94	37.31	**13.58**
TouchDown	TouchDown	0-pact-geo-a	87.66	77.32	87.59
		base-pg-a	**67.54**	**64.44**	**72.07**
		0-pact-geo-b	95.6	90.75	97.18
		base-pg-b	**66.77**	**67.9**	**74.21**
	Map2Seq	0-pact-geo-a	**78.12**	**79.03**	94.91
		base-pg-a	87.41	84.9	**89.59**
		0-pact-geo-b	87.95	89.2	98.57
		base-pg-b	**85.17**	**83.52**	**90.37**
Map2Seq	TouchDown	0-pact-geo-a	60.2	56.15	68.24
		base-pg-a	**45.27**	**33.02**	**45.02**
		0-pact-geo-b	83.4	70.1	92.42
		base-pg-b	**34.34**	**45.8**	**28.85**
	Map2Seq	0-pact-geo-a	**47.47**	**42.22**	73.2
		base-pg-a	48.35	51.43	**61.3**
		0-pact-geo-b	62.03	61.46	78.25
		base-pg-b	**54.55**	**52.48**	**50**

Table 15: Overshoot Rate (OSR) among different models fine-tuned without vision and data splits. For each pair of Zero PActs and Geographical Overlap (0-pact-geo-x) and control splits (base-pg-x), the best performing split is bolded

Test Dataset	Finetune Dataset	Image	Split	Llama1-7B			Llama2-7B			Mistral-7B		
				TC	OSH	OSR	TC	OSH	OSR	TC	OSH	OSR
TouchDown	TouchDown	No-Vision	base-p	**13.53**	24.72	**64.62**	**15.58**	20.32	**56.61**	**8.06**	24.48	**75.23**
			0-pact	7.68	**23.65**	75.49	7.77	**12.12**	60.92	3.05	**10.52**	77.51
		OpenCLIP	base-p	**28.22**	**31.91**	**53.07**	**28.34**	**24.53**	**46.4**	**14.92**	35.57	**70.45**
			0-pact	18.69	37.48	66.72	24.28	29.5	54.85	7.05	**27.17**	79.41
	Map2Seq	No-Vision	base-p	**5.97**	21.78	**78.49**	**5.08**	24.46	82.82	**2.96**	27.26	**90.2**
			0-pact	3.17	**13.22**	80.66	3.55	**14.09**	**79.89**	1.36	**16.25**	92.27
		OpenCLIP	base-p	**7.82**	23.19	**74.77**	**7.31**	23.12	**75.98**	**5.83**	27.02	**82.26**
			0-pact	3.76	**13.48**	78.2	4.56	**14.49**	76.08	2.02	**17.24**	89.51
Map2Seq	TouchDown	No-Vision	base-p	**27.24**	22.46	45.2	**29.51**	12.63	29.97	**9.49**	7.19	**43.1**
			0-pact	20.59	**15.89**	**43.56**	25.32	**10.38**	**29.08**	2.74	**3.46**	55.81
		OpenCLIP	base-p	**31.64**	14.55	31.51	**27.7**	6.79	19.68	**8.57**	6.61	**43.56**
			0-pact	18.72	**7.67**	**29.07**	22.24	**4.7**	**17.45**	3.52	**3.52**	50
	Map2Seq	No-Vision	base-p	**40.98**	26.6	39.36	**37.1**	33.42	**47.39**	**25.65**	**36.93**	**59.01**
			0-pact	38.13	**24.4**	**39.02**	34.7	**31.55**	47.63	21.81	41.16	65.37
		OpenCLIP	base-p	**50.1**	**16.42**	**24.69**	**50.16**	**18.38**	**26.81**	**39.57**	**25.54**	**39.22**
			0-pact	46.15	16.56	26.4	43.01	23.8	35.62	35.13	27.34	43.77

Table 16: Overshoot (OSH) denotes the number of overshoot cases among all of the samples in the test split. Overshoot Rate (OSR) and Task Completion (TC) are described in section 5.5. As explained in section 5.5, separation of PActs from train and test, results in lower number of OSH cases and TC rates in 0-pact compared to its baseline, base-p. However, the overall outcome is a general increase in Overshoot rates.

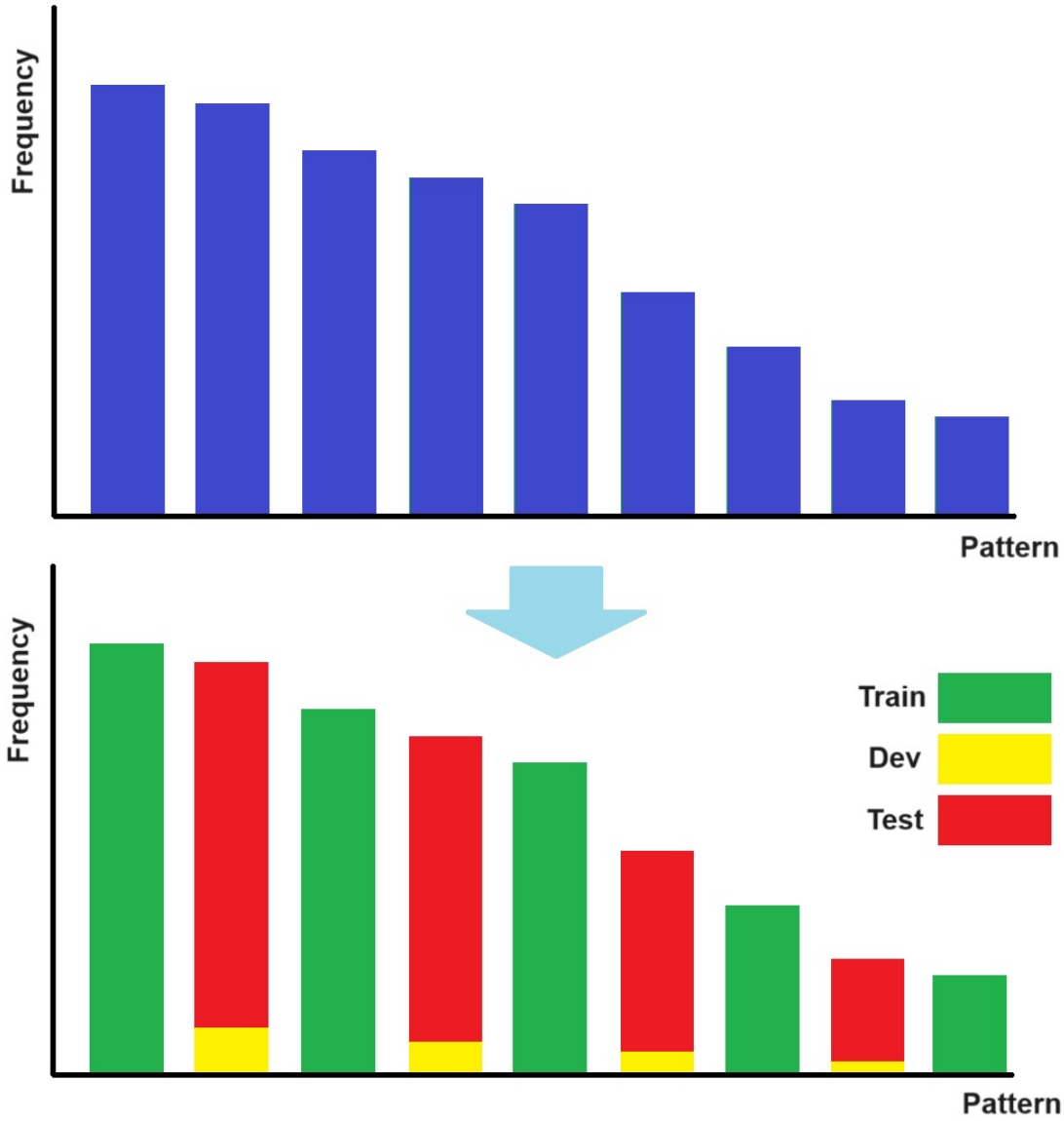


Figure 2: Illustration of creation of Zero Pattern Overlap from base-unseen split. The graphs depicted here are hypothetical to clarify the process. Each column represents frequency (number of repetitions) of a pattern in samples. Splitting the data by patterns, results in zero pattern overlap, whereas geographical overlap still exists.

D Data Augmentation

Pattern Generation The structure of patterns can be defined as a regex as follows. For Touch-Down:

$$[t|l|r]?(fl|fr)*(fs) \quad (1)$$

And for Map2Seq dataset, where the initial direction of the agent is towards the ground truth path:

$$(fl|fr)*(fs) \quad (2)$$

Where f,l,r,t,s correspond to {`FORWARD`, `LEFT`, `RIGHT`, `TURN_AROUND`, `STOP`} actions respectively. We create all possible combinations of patterns with up to 7 turns meaning that a pattern would contain 7 intermediate turns. This results in 127 unique patterns for M2S, and 508 unique patterns for TD. For instance, the following patterns contain 3 turns: *flflfrfs, lfrflfrfs*

Instruction Generation Once we have a pattern in hand, we create instructions that correspond to each action using templates with blanks reserved for landmarks. For instance: *'Turn so [blank] is on your [blank]'* where we can fill them with *Starbucks, left* respectively. We concatenate the filled templates to form a piece of navigational instructions.

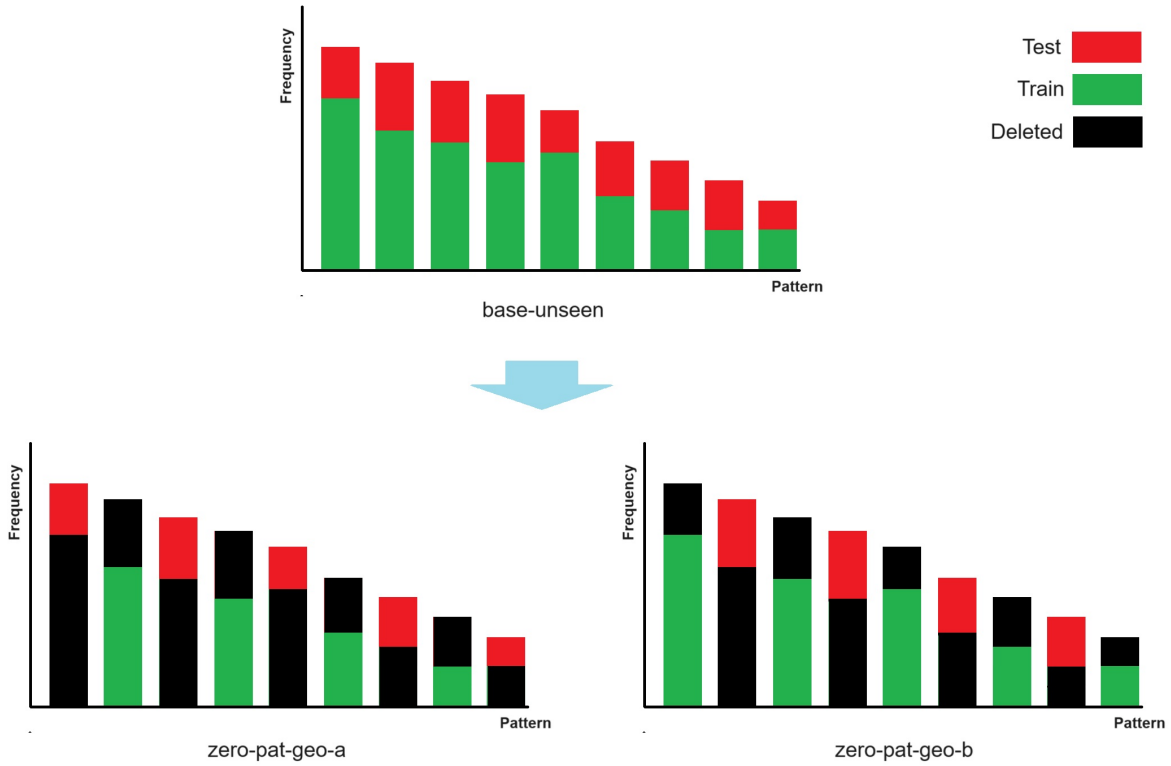


Figure 3: Illustration of creation of Zero Pattern and Geographical Overlap from base-unseen split. The graphs depicted here are hypothetical to clarify the process. Each column represents frequency (number of repetitions) of a pattern in samples. In base unseen, train and test samples are geographically separate. So, when we separate them by patterns, we could get two sub-sets that are (a) geographically separate, AND (b) have zero pattern overlap. From the samples assigned to the test, we randomly take 15 (20) percent of samples to create dev set for TouchDown (Map2Seq) and use the remaining samples as the test set.

Training Sample Following (Schumann et al., 2024), each piece of instructions is placed within a prompt template, that instructs the agent to predict the next action given the instructions and previous sequence of actions. So, to create a training sample, we also create the ground truth sequence of actions and observations that correspond to the pattern. For instance for a pattern: *flfs* where the instruction is: *Go straight until you reach Salsa Room, then turn left. Keep going straight until you reach Apple store* The sequence of actions and observations would look like as follows: *1.Forward, 2.Forward, 3. Forward, Salsa Room on your right, 4. Left, 5. Forward, 6. Forward, Apple store ahead, 7. Stop*

Avoiding Over-fitting To avoid over-fitting to the templates, we take the following measures:

1. We paraphrase the instructions using GPT-4o using the prompt in Figure 5.
2. During the training we add noises to the dataset in three ways:
   - *Word Drop:* we randomly drop 10 percent of the words in the instructions
   - *Letter Drop:* we randomly drop one letter from 10 percent of the remaining words
   - *Added Noise:* We repeat up to 50 percent of the observations several times among a sequence of actions. This is a type of noise that can happen in real data. For instance when an agent passes by a bank, it might be able to see the bank within several consequent steps. So, the observation will be added to the prompt several times.

Data Generation Settings We create dataset with the following different settings.

1. **Landmarks from Human Generated Data** When creating a synthetic dataset, we randomly choose from landmarks referred in the human generated dataset.

2. **Landmarks from New Town** In another setting, we create landmarks that are totally synthetic and imaginary i.e. formed by putting

random words together. For instance *Bright Mountain Cafe*.

3. **Zero Initial Turns** We also create another dataset where the initial action is never a turning action i.e. *left, right, turn_around*. In this setting, the pattern

Fine-tuning Table 17 shows how different factors affect the performance of a model. Sampling landmarks from the current human generated data to create new synthetic routes might result in over-fitting of the model. So, we articulate that one way of avoiding over-fitting would be to generate synthetic landmarks. This approach, improves performance of the model in certain cases.

Another problem with the current setting of VLN agent is the ability of the agent in alignment of landmarks. This causes the failure of the model to turn appropriately in the beginning of a route. So, we articulate that training on more samples of data, would not contribute much to the initial turn task as long as the model does not ground well the visual observations. So, we create a dataset where none of its patterns start with a turning task. However, such a change, does not solely contribute to any better performance.

And lastly, any human generated data may contain errors such as missing letters in words or even missing words in a sentence. So, we also add such noises to the data during the training phase. So that the noise is applied at random and in many different combinations.

Combining all of the above strategies, results in maximum enhancements given our settings. However, we leave a more comprehensive study of data augmentation methods as a future work.

E Instruction Templates

Figure 4 shows the raw instructions templates used for data generation.

F Prompts

Figure 5 shows the prompt used for paraphrasing the raw template-based instructions.

		Synthetic data settings				
	Baseline	✓				
	New Town		✓			✓
	0 Inital Turns			✓		✓
	Added Noise				✓	✓
FT → Test						
Same train-test datasets						
TD → TD	No-Vision	13.8	**13.93**	10.06	13.33	13.13
	OpenCLIP	23.22	**23.8**	12.87	15.12	22.56
M2S → M2S	No-Vision	26.5	22.5	22.12	31.4	**31.5**
	OpenCLIP	36.75	30.87	33.25	**37.6**	**39.25**
Different train-test datasets						
TD → M2S	No-Vision	3.58	3.26	3.13	3.4	**3.8**
	OpenCLIP	4.98	4.33	4.46	4.66	**5.1**
M2S → TD	No-Vision	25.5	19.12	20.37	22.6	**26.1**
	OpenCLIP	23	19.75	21.25	22.25	**23.5**

Table 17: TC rate (%) for different settings of data augmentation for Llama-2. Any number in bold shows where the agent outperforms the baseline.

Initial direction with respect to surrounding landmarks:
Turn so [landmark] is on your [direction]

Initial Direction with respect to the traffic direction:
Orient yourself [against|with] the flow of the traffic
Orient yourself so you are facing [against|with] the flow of the traffic

Moving forward:
Go straight
Go straight until you [reach|see] [landmark]

Making a turn:
Turn [direction] at the [traffic light|intersection]
Turn [direction] when you see [landmark] on your [direction]
At the [intersection|street light] that you see [landmark] on your [direction], turn [direction]

Stop:
Stop when you see [landmark]
Stop at the [landmark]
Stop when you see [landmark] on your [direction]
Stop

Observation:
There is a [3|4]-way intersection.
You are aligned [against|with] the flow of the traffic

Figure 4: Raw templates of synthetic instructions. Place holders indicated with brackets are replaced with proper words.

Initial direction with respect to surrounding landmarks:
Rephrase the given text so that its meaning remains the same by using a combination of the following approaches:

1. make it more formal 2. make it less formal 3. change the words with appropriate synonyms 4. combine some of the sentences together. 5. split long sentences into two sentences. 6. move around the verb and objective within a sentence.

Only write the resulting text.

The given text:

[instructions]

Figure 5: Prompt for rephrasing the template based .