On the Evaluation of Vision-and-Language Navigation Instructions

Ming Zhao Peter Anderson Vihan Jain Su Wang
Alexander Ku Jason Baldridge Eugene Ie
Google Research
{astroming, pjand, vihan, wangsu, alexku, jridge, eugeneie}
@google.com

Abstract

Vision-and-Language Navigation wayfinding agents can be enhanced by exploiting automatically generated navigation instructions. However, existing instruction generators have not been comprehensively evaluated, and the automatic evaluation metrics used to develop them have not been validated. Using human wayfinders, we show that these generators perform on par with or only slightly better than a template-based generator and far worse than human instructors. Furthermore, we discover that BLEU, ROUGE, METEOR and CIDEr are ineffective for evaluating grounded navigation instructions. To improve instruction evaluation, we propose an instruction-trajectory compatibility model that operates without reference instructions. Our model shows the highest correlation with human wayfinding outcomes when scoring individual instructions. For ranking instruction generation systems, if reference instructions are available we recommend using SPICE.

1 Introduction

Generating route instructions is a long studied problem with clear practical applications (Richter and Klippel, 2005). Whereas earlier work sought to create instructions for human wayfinders, recent work has focused on using instruction-generation models to improve the performance of agents that follow instructions given by people. In the context of Vision-and-Language Navigation (VLN) datasets such as Room-to-Room (R2R) (Anderson et al., 2018b), models for generating navigation instructions have improved agents' wayfinding performance in at least two ways: (1) by synthesizing new instructions for data augmentation (Fried et al., 2018; Tan et al., 2019), and (2) by fulfilling the role of a probabilistic speaker in a pragmatic reasoning setting (Fried et al., 2018). Such data augmentation is so effective that it is nearly ubiquitous in the best performing agents (Wang et al., 2019; Huang et al., 2019; Li et al., 2019).

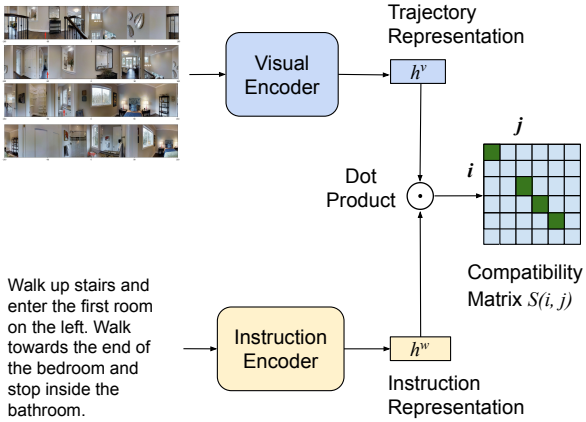

Figure 1: Proposed dual encoder instruction-trajectory compatibility model. Navigation instructions and trajectories (sequences of panoramic images and view angles) are projected into a shared latent space. The independence between the encoders facilitates learning using both contrastive and classification losses.

To make further advances in the generation of visually-grounded navigation instructions, accurate evaluation of the generated text is essential. However, the performance of existing instruction generators has not yet been evaluated using human wayfinders, and the efficacy of the automated evaluation metrics used to develop them has not been established. This paper addresses both gaps.

To establish benchmarks for navigation instruction generation, we evaluate existing English models (Fried et al., 2018; Tan et al., 2019) using human wayfinders. These models are effective for data augmentation, but in human trials they perform on par with or only slightly better than a template-based system, and they are far worse than human instructors. This leaves much headroom for better instruction generation, which may in turn improve agents' wayfinding abilities.

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pages 1302–1316
April 19 - 23, 2021.

Next, we consider the evaluation of navigation instructions without human wayfinders, a necessary step for future improvements in both grounded instruction generation (itself a challenging and important language generation problem) and agent wayfinding. We propose a model-based approach (Fig. 1) to measure the compatibility of an instruction-trajectory pair without needing reference instructions for evaluation. In training this model, we find that adding contrastive losses in addition to pairwise classification losses improves AUC by 9–10%, round-trip back-translation improves performance when used to paraphrase positive examples, and that both trajectory and instruction perturbations are useful as hard negatives.

Finally, we compare our compatibility model to common textual evaluation metrics to assess which metric best correlates with the outcomes of human wayfinding attempts. We discover that BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Denkowski and Lavie, 2014) and CIDEr (Vedantam et al., 2015) are ineffective for evaluating grounded navigation instructions. For system-level evaluations with reference instructions, we recommend SPICE (Anderson et al., 2016). When averaged over many instructions, SPICE correlates with both human wayfinding performance and subjective human judgments of instruction quality. When scoring individual instructions, our compatibility model most closely reflects human wayfinding performance, outperforming BERTScore (Zhang et al., 2019) and VLN agent-based scores. Our results are a timely reminder that textual evaluation metrics should always be validated against human judgments when applied to new domains. We plan to release our trained compatibility model and the instructions and human evaluation data we collected.

2 Related Work

Navigation Instruction Generation Until recently, most methods for generating navigation instructions were focused on settings in which a system has access to a map representation of the environment, including the locations of objects and named items (e.g. of streets and buildings) (Richter and Klippel, 2005). Some generate route instructions *interactively* given the current position and goal location (Dräger and Koller, 2012), while others provide *in-advance* instructions that must be more robust to possible misinterpretation (Roth and Frank, 2010; Mast and Wolter, 2013).

Recent work has focused on instruction generation to improve the performance of wayfinding agents. Two instruction generators, *Speaker-Follower* (Fried et al., 2018) and *EnvDrop* (Tan et al., 2019), have been widely used for R2R data augmentation. They provide ∼170k new instruction-trajectory pairs sampled from training environments. Both are seq-to-seq models with attention. They take as input a sequence of panoramas grounded in a 3D trajectory, and output a textual instruction intended to describe it.

Vision-and-Language Navigation For VLN, embodied agents in 3D environments must follow natural language instructions to reach prescribed goals. Most recent efforts (e.g., Fu et al., 2019; Huang et al., 2019; Jain et al., 2019; Wang et al., 2019, etc.) have used the Room-to-Room (R2R) dataset (Anderson et al., 2018b), which contains 4675 unique paths in the *train* split, 340 in the *val-seen* split (same environments, new paths), and an additional 783 paths in the *val-unseen* split (new environments, new paths). However, our findings are also relevant for similar datasets such as Touchdown (Chen et al., 2019; Mehta et al., 2020), CVDN (Thomason et al., 2019), REVERIE (Qi et al., 2020), and the multilingual Room-across-Room (RxR) dataset (Ku et al., 2020).

Text Generation Metrics There are many automated metrics that assess textual similarity; we focus on five that are extensively used in the context of image captioning: BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), ROUGE (Lin, 2004), CIDEr (Vedantam et al., 2015) and SPICE (Anderson et al., 2016). More recently, model- and semi-model-based metrics have been proposed. BERTScore (Zhang et al., 2019) takes a semi-model-based approach to compute token-wise similarity using contextual embeddings learned with BERT (Devlin et al., 2019). BLEURT (Sellam et al., 2020) is a fully model-based approach combining large-scale synthetic pretraining and domain specific finetuning. However, all of the aforementioned metrics are reference-based, and none is specifically designed for assessing navigation instructions associated with 3D trajectories for an embodied agent, which requires not only language-to-vision grounding but also correct sequencing.

Instruction-Trajectory Compatibility Models Our model builds on that of Huang et al. (2019), but differs in loss (using focal and contrastive

losses), input features (adding action and geometry representation), and negative mining strategies (adding instruction perturbations in addition to trajectory perturbations). Compared to the trajectory re-ranking compatibility model proposed by Majumdar et al. (2020), we use a dual encoder architecture rather than dense cross-attention. This facilitates the efficient computation of contrastive losses, which are calculated over all pairs in a mini-batch, and improve AUC by 10% in our model. We also avoid training on the outputs of the instruction generators (to prevent overfitting to the models we evaluate). We are yet to explore transfer learning (which is the focus of Majumdar et al. (2020)).

3 Human Wayfinding Evaluations

To benchmark the current state-of-the-art for navigation instruction generation, we evaluate the outputs of the *Speaker-Follower* and *EnvDrop* models by asking people to follow them. We use instructions for the 340 and 783 trajectories in the R2R val-seen and val-unseen splits, respectively. Both models are trained on the R2R train split and the generated instructions were provided by the respective authors. To contextualize the results, we additionally evaluate instructions from a template-based generator (using ground-truth object annotations), a new set of instructions written by human annotators, and three adversarial perturbations of these human instructions. New navigation instructions and wayfinding evaluations are collected using a lightly modified version of PanGEA[1], an open-source annotation toolkit for panoramic graph environments.

Crafty Crafty is a template-based navigation instruction generator. It observes the trajectory's geometry and nearby ground-truth object annotations, identifies salient objects, and creates English instructions using templates describing movement with respect to the trajectory and objects. See the Appendix for details. Note that Crafty has an advantage over the learned models which rely on panoramic images to identify visual references and do not exploit object annotations.

Human Instructions We collect 340 new English instructions for the trajectories in the R2R val-seen split using the PanGEA Guide task.

Instruction Perturbations To quantify the impact of common instruction generator failure modes on instruction following performance, we include three adversarial perturbations of human instructions capturing incorrect direction words, hallucinated objects/landmarks, and repeated or skipped steps. We use Google Cloud NLP[2] to identify named entities and parse dependency trees and then generate perturbations as follows:

- **Direction Swap**: Random swapping of directional phrases with alternatives from the same set, with sets as follows: *around/left/right, bottom/middle/top, up/down, front/back, above/under, enter/exit, backward/forward, away from/towards, into/out of, inside/outside*.

  Example: *"Take a right (**left**) and wait by the couch outside (**inside**) the bedroom. "*

- **Entity Swap**: Random swapping of entities in an instruction. All noun phrases excluding a stop list containing *any, first, end, front*, etc. are considered to be entities. If two entities have the same lemma (e.g., *stairs/staircase/stairway*) they are considered to be synonyms and are not swapped.

  Example: *"Exit the bedroom (**bathroom**), turn right, then enter the bathroom (**bedroom**)."*

- **Phrase Swap**: A random operation on the dependency tree: either remove one sub-sentence tree, duplicate one sub-sentence tree, or shuffle the order of all sentences except the last.

  Example: *"Exit the room using the door on the left. Turn slightly left and go past the round table an chairs. Wait there."* – where the first and second sentences are swapped.

Wayfinding Task Using the PanGEA Follower task, annotators are presented with a textual navigation instruction and the first-person camera view from the starting pose. They are instructed to attempt to follow the instruction to reach the goal location. Camera controls allow for continuous heading and elevation changes as well as movement between Matterport3D panoramas based on a navigation graph. Each instruction is evaluated by three different human wayfinders.

[1] https://github.com/google-research/pangea

[2] https://cloud.google.com/natural-language/

Instructions	Num. Evals	Wordcount	NE ↓	SR ↑	SPL ↑	SDTW ↑	Quality ↑	Visual Search		Time ↓
								Start ↓	Other ↓	
Val-unseen										
Speaker-Follower	783×3	24.6	6.55	35.8	30.3	28.1	3.50	43.5	24.8	43.2
EnvDrop	783×3	21.3	5.89	42.3	36.1	33.5	3.70	42.1	24.8	**39.5**
Val-seen										
Speaker-Follower	340×3	24.7	6.23	42.3	35.7	33.2	3.64	43.0	24.5	42.6
EnvDrop	340×3	22.3	5.99	47.7	40.0	36.9	3.83	39.9	25.0	42.1
Crafty	340×3	71.2	6.01	43.6	34.7	33.3	3.48	42.0	25.9	69.6
Direction Swap	340×3	54.8	4.74	58.9	47.9	45.9	3.67	40.6	25.4	61.0
Entity Swap	340×3	55.1	4.71	51.3	42.5	40.6	3.33	40.1	25.8	62.7
Phrase Swap	340×3	52.0	4.07	62.6	51.6	49.9	3.85	38.7	24.7	58.0
Human	340×3	54.1	**2.56**	**75.1**	**64.7**	**63.1**	**4.25**	**35.8**	**23.8**	53.9

Table 1: Human wayfinding performance following instructions from the Speaker-Follower (Fried et al., 2018) and EnvDrop (Tan et al., 2019) models, compared to Crafty (template-based) instructions, Human instructions, and three adversarial perturbations of Human instructions (Direction, Entity and Phrase Swap).

Evaluation Metrics We use the following standard metrics to evaluate the trajectories generated by our annotators (and hence, the quality of the provided instructions): *Navigation Error* (**NE** ↓) *Success Rate* (**SR** ↑), *Success weighted by inverse Path Length* (**SPL** ↑), *Success weighted by normalized Dynamic Time Warping* (**SDTW** ↑). Arrows indicate improving performance. See Anderson et al. (2018a) and Ilharco et al. (2019) for details.

People are resourceful and may succeed in following poor quality instructions by expending additional effort. Therefore, we report additional metrics to capture these costs. **Quality** ↑ is a self-reported measure of instruction quality based on a 1–5 Likert scale. At the end of each task annotators respond to the prompt: *Do you think there are mistakes in the instruction?* Responses range from *Way too many mistakes to follow (1)* to *No mistakes, very very easy to follow' (5)*. **Visual Search** cost ↓ measures the percentage of the available panoramic visual field that the annotator observes at each viewpoint, based on the pose traces provided by PanGEA and first proposed in the RxR dataset (Ku et al., 2020). Higher values indicate greater effort spent looking for the correct path. We report this separately for the start viewpoint and other viewpoints since wayfinders typically look around to orient themselves at the start. **Time** ↓ represents the average time taken in seconds.

Results Table 1 summarizes the results of 11,886 wayfinding attempts using 37 English-speaking annotators. The performance of annotators stays consistent over time and does not show any sign of adaptation. See Appendix for detailed analysis.

As expected, human instructions perform best in human wayfinding evaluations on all path evaluation metrics and on subjective assessments of instruction quality, and they also incur the lowest visual search costs. The only metric not dominated by human instructions is the time taken – which correlates with instruction length, and may be affected by wayfinders giving up when faced with poor quality instructions. Overall, the Speaker-Follower and EnvDrop models are surprisingly weak and noticeably worse than even adversarially perturbed human instructions. Compared to the template-based approach (Crafty), the Speaker-Follower model performs on par and EnvDrop is only slightly better. As a first step to improving existing navigation instruction generators, we focus on developing and evaluating automated metrics that can approximate these human wayfinding evaluations.

4 Compatibility Model

As an alternative to human evaluations, we train an instruction-trajectory compatibility model to assess both the grounding between textual and visual inputs and the alignment of the two sequences.

4.1 Model Structure

Our model is a dual encoder that encodes instructions and trajectories into a shared latent space (Figure 1). The instruction representation h^w is the concatenation of the final output states of a bi-directional LSTM (Schuster and Paliwal, 1997) encoding the instruction tokens $\mathcal{W} = \{w_1, w_2, ..., w_n\}$. We use contextualized token embeddings from BERT (Devlin et al., 2019) as input to the LSTM.

The visual encoder is a two-layer LSTM that processes visual features extracted from a sequence

of viewpoints $\mathcal{V} = \{(I_1, p_1), (I_2, p_2)..., (I_t, p_t)\}$ comprised of panoramic images I_t captured at positions p_t along a 3D trajectory. The vector h_t^v representing the viewpoint at step t is given by:

$$a_t = \text{Attention}(h_{t-1}^v, e_{\text{pano},t}) \quad (1)$$

$$v_t = f([e_{\text{prev},t}, e_{\text{next},t}, a_t]) \quad (2)$$

$$h_t^v = \text{LSTM}(v_t, h_{t-1}^v) \quad (3)$$

where $e_{\text{pano},t}$ is a set of 36 visual features representing the panoramic image I_t (discretized into 36 viewing angles by elevation θ and heading ϕ), $e_{\text{prev},t}$ and $e_{\text{next},t}$ are the visual features in the directions of the previous and next viewpoints ($v_{t-1} - v_t$ and $v_{t+1} - v_t$ respectively), and f is a projection layer. Each visual feature is a concatenation of a pre-trained CNN image feature (Juan et al., 2020) with orientation vectors encoding both sine and cosine functions of the absolute and relative angles $\{\theta_{\text{abs}}, \phi_{\text{abs}}, \theta_{\text{rel}}, \phi_{\text{rel}}\}$. We use standard dot-product attention (Luong et al., 2014) and define $h^v = h_T^v$, the final viewpoint embedding in the trajectory. The output of the model is the compatibility score S between an instruction and a trajectory defined as the cosine similarity between h^v and h^w.

4.2 Hard Negative Mining

To avoid overfitting, our compatibility model is *not trained on the outputs of any of the instruction generators that we evaluate*. Instead, we use only the relatively small set of positive instruction-trajectory examples from R2R. We use round-trip back-translation to expand the set of positive examples. Unmatched instruction-trajectory pairs from R2R are considered to be negative examples and we also construct hard negative examples from positive examples by adversarially perturbing both trajectories and instructions.

Instruction Perturbations We use the same instruction perturbations described in Section 3: *Direction Swap*, *Entity Swap*, and *Phrase Swap*. These perturbations are inspired by typical failure modes in instruction generators and are designed to be hard to recognize without grounding on images and actions along the trajectory. Previous work by Huang et al. (2019) considered only trajectory perturbations. While this encourages the model to recognize incorrect trajectories for a given ground truth instruction, it may not encourage the model to identify a trajectory matched with a poor quality instruction. Our results suggest that instruction perturbations are equally important.

Trajectory Perturbations To perturb trajectories we use the navigation graphs defining connected viewpoints in R2R. Inspired by Huang et al. (2019), we consider *Random Walk*, *Path Reversal*, and *Viewpoint Swap* perturbations:

- **Random Walk**: The first or last two viewpoints are fixed and the remainder of the trajectory is re-sampled using random edge traversals subject to the path length remaining within ± 1 step of the original. To make the task harder, we avoid revisiting a viewpoint and require the re-sampled trajectory to have at least two overlapping viewpoints with the original.

- **Path Reversal**: The entire trajectory is reversed while keeping the same viewpoints.

- **Viewpoint Swap**: A new method we introduce that randomly samples and swaps a viewpoint in a trajectory with a new viewpoint sampled from the neighbors of the adjacent viewpoints in the original trajectory.

Paraphrases To expand the 14k positive examples from the R2R train set and balance the positive-to-negative ratio, we paraphrase instructions via round-trip back-translation. We use the following ten intermediate languages and Google Translate[3]: *ar, es, de, fr, hi, it, pt, ru, tr*, and *zh*. To exclude low quality or nearly duplicate instructions, we filter paraphrased instructions outside the BLEU score range of [0.25, 0.7] compared to the original. Overall we have a total of 110,601 positive instruction-trajectory pairs in the training set, which contains 4675 unique trajectories.

4.3 Loss Functions

During training, each minibatch is constructed with N matching instruction-trajectory pairs, which may be perturbed. We define $M \in \{0, 1\}^N$ as the vector indicating unperturbed pairs. A compatibility matrix $S \in \mathcal{R}^{N \times N}$ is defined such that $S_{i,j}$ is the cosine similarity score between instruction i and trajectory j determined by our model.

We use both binary classification loss functions, defined on diagonal elements of S, and a contrastive loss defined on S's rows and columns. Contrastive losses are commonly used for retrieval and representation learning (e.g., Yang et al., 2019;

[3] https://cloud.google.com/translate/

	Model	Perturbations	Validation (val_unseen)	Test (val_seen)	Instruction Validation			Path Validation		
					Direction Swap	Entity Swap	Phrase Swap	Viewpoint Swap	Random Walk	Path Reversal
1	Huang et al. (2019)	Path + Instruction	53.4	52.3	80.3	89.4	80.9	78.0	72.9	75.9
	This Work									
2	+ CE Loss	Path + Instruction	57.9	57.6	89.5	88.4	83.4	95.0	94.1	87.8
3	+ Focal Loss	Path + Instruction	59.2	59.2	**89.8**	90.5	84.2	95.6	95.0	90.3
4	+ Contrastive Loss	Path + Instruction	67.2	68.7	75.1	70.1	73.7	73.6	88.7	93.1
5	+ Contrastive + CE	Path + Instruction	66.5	67.5	82.0	77.4	76.9	84.7	91.3	91.7
6	+ Contrastive + Focal	Path + Instruction	68.5	68.3	83.9	81.5	79.8	88.5	93.7	93.3
7	+ Contrastive + Focal + Paraphrase	Path + Instruction	71.3	72.2	83.1	81.9	80.4	90.4	95.2	**94.0**
8	+ Contrastive + Focal + Paraphrase + Bert Embed.	Path + Instruction	**73.5**	**73.7**	84.6	86.7	82.2	89.1	94.3	93.3
	Perturbation Ablations									
9	+ Contrastive + Focal + Paraphrase + Bert Embed.	Instruction	70.5	70.4	85.1	88.6	82.5	64.9	84.3	90.7
10	+ Contrastive + Focal + Paraphrase + Bert Embed.	Direction Swap Only	70.1	68.9	89.0	72.0	70.7	65.9	85.0	91.0
11	+ Contrastive + Focal + Paraphrase + Bert Embed.	Entity Swap Only	69.1	68.5	70.3	**92.7**	71.3	65.2	84.4	90.9
12	+ Contrastive + Focal + Paraphrase + Bert Embed.	Phrase Swap Only	70.3	70.5	70.1	72.5	**85.5**	65.5	83.3	90.1
13	+ Contrastive + Focal + Paraphrase + Bert Embed.	Path	69.7	69.4	71.7	71.4	69.8	92.4	94.8	92.2
14	+ Contrastive + Focal + Paraphrase + Bert Embed.	Viewpoint Swap Only	70.7	72.1	70.7	71.1	71.2	**94.6**	94.6	91.9
15	+ Contrastive + Focal + Paraphrase + Bert Embed.	Random Walk Only	70.5	70.9	70.5	71.5	71.0	81.7	**95.1**	91.6
16	+ Contrastive + Focal + Paraphrase + Bert Embed.	Path Reversal Only	69.4	69.7	71.0	70.7	71.2	65.3	86.3	91.9
17	+ Contrastive + Focal + Paraphrase + Bert Embed.	No Perturbation	69.1	69.7	71.1	71.0	70.8	65.3	84.9	90.2

Table 2: Ablation of different models based on classification AUC. Models are trained with the original R2R-train data and paraphrased positive instructions, plus path and/or instruction perturbed hard negatives (second column). The best models are selected based on the validation set (column 3), and we report the final test performance in column 4. To understand the performance of individual perturbation method, we also report the best AUCs for each of the six perturbations in columns 5 - 10.

Chen et al., 2020) and in our case exploits all random instruction-trajectory pairs in a minibatch.

Each loss requires a separate normalization. For the classification loss we compute the probability of a match $p_{i,j}$, such that $p_{i,j} = \sigma(aS_{i,j} + b)$ where a and b are learned scalars and σ is the sigmoid function. For the classification loss $\mathcal{L}_{\text{cls}}$ we consider both binary cross entropy loss $\mathcal{L}_{\text{CE}}$, and focal loss (Lin et al., 2017) given by $\mathcal{L}_{\text{FL}} = (1 - p_{i,j})^{\gamma}\mathcal{L}_{\text{CE}}$ where we set $\gamma = 2$.

For the contrastive loss we compute logits by scaling S with a learned scalar temperature τ. The contrastive loss $\mathcal{L}_C(S)$ calculated over the rows and colums of S is given by:

$$\mathcal{L}_C(S) = \frac{1}{\sum_i M_i} \sum_{i=1}^{N} \left(\mathcal{L}_r(S_i) + \mathcal{L}_r(S_i^{\intercal}) \right) \quad (4)$$

where $\mathcal{L}_r(S_i) = 0$ if $M_i = 0$, i.e., the diagonal element is a perturbed pair and not considered to be a match. Otherwise:

$$\mathcal{L}_r(S_i) = -\log \frac{e^{S_{i,i}/\tau}}{\sum_j^N e^{S_{i,j}/\tau}} \quad (5)$$

The final loss is the combination:

$$\mathcal{L} = \mathcal{L}_C(S) + \frac{\beta}{N} \sum_{i=1}^{N} \mathcal{L}_{cls}(S_{i,i}) \quad (6)$$

where $\mathcal{L}_{cls}$ is the classification loss, either $\mathcal{L}_{CE}$ or $\mathcal{L}_{FL}$, and we set $\beta = 1$.

Sampling hyperparameters We sample positive and negative examples equally with a mix ratio of 2:1:1 for ground truth, instruction perturbations, and trajectory perturbations, respectively. For each perturbation type, we sample the three methods with equal probability.

5 Experiments

We evaluate our compatibility model against alternative model-based evaluations and standard textual similarity metrics. We report instruction classification results in Section 5.1, improved data augmentation for VLN agents in Section 5.2, and correlation with human wayfinder outcomes in 5.3.

5.1 Instruction Classification

Evaluation In this setting we use the instruction-trajectory compatibility model to classify high and low quality instructions for trajectories from the R2R val-unseen and val-seen sets. The instruction pool includes 3 high-quality instructions per trajectory from R2R, plus 2 instructions per trajectory from the Speaker-Follower and EnvDrop models. These are considered to be high quality if 2 out of 3 human wayfinders reached the goal (see Section 3), and low quality otherwise. We assess model performance using *Area Under the ROC Curve* (AUC). We use the val-unseen split (3915 instructions, 75% high quality) for model validation and the val-seen split (1700 instructions, 78% high quality) as test.

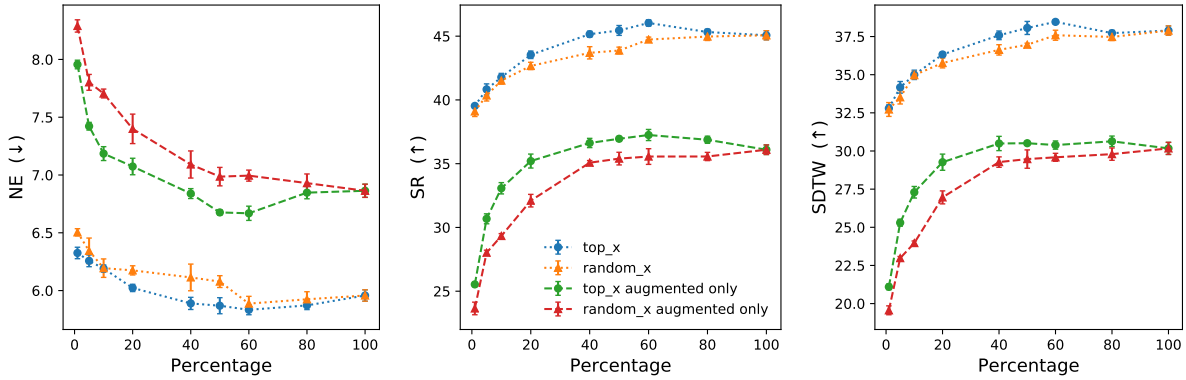

Figure 2: Navigation performance of a VLN agent trained with different fractions of Speaker-Follower augmented paths, starting from 1%. For NE, lower is better; for SR and SDTW, higher is better. The dashed-lines (green and red) use only augmented paths in training, while the dotted lines (blue and orange) use both augmented and R2R-train paths. Filled circles indicate fractions ranked by our compatibility model in descending order, while triangles indicate random fractions. Each point is the mean of 3 runs and the error bars represent the standard deviation of the mean. The model-ranked fractions show consistent improvement over random samples of the same percentage. Agents trained with only augmented paths (dashed-lines) show greater difference between model-ranked fractions and random fractions.

Benchmark We compare to the compatibility model proposed by Huang et al. (2019), which computes elementwise similarities between instruction words and trajectory panoramas *before* pooling, and does not include action embeddings ($e_{\text{prev},t}$ and $e_{\text{next},t}$) or position encodings p_t. In contrast, our model calculates the similarity between instructions and trajectories *after* pooling each sequence and includes both action and position encodings.

Results Table 2 reports classification AUC including comprehensive ablations of loss functions, approaches to hard negative mining, and modeling choices. With regard to the loss function, we find that the combination of contrastive and focal loss (row 6) performs best overall, and that adding contrastive loss provides a very significant 9 - 10% increase in AUC compared to using just cross-entropy (CE) or focal loss (rows 2 and 3) due to the effective use of in-batch negatives. Adding paraphrased positive instructions and pretrained BERT token embeddings also leads to significant performance gains (rows 7 and 8 vs. row 6). The best performing model on both the validation and test sets uses Contrastive + Focal loss with paraphrased instructions and BERT embeddings, as well as trajectory and instruction perturbations (row 8). This model consistently outperforms the benchmark from prior work (row 1) by a large margin and achieves a test set AUC of 73.7%.

In rows 9–17 we ablate the six perturbation methods that we use for hard negative mining. Ablations using only instruction perturbations (row 9), only path perturbations (13), or no perturbations at all (row 17) perform considerably worse than our best model (row 8). We also show that no individual perturbation approach is effective on its own. In addition to scores for the validation and test sets, we report AUC for each perturbation method on the val-seen set to investigate their individual performance. Overall, trajectory perturbations get higher scores than instruction perturbations, showing they are easier tasks. *Phrase Swap* proves the hardest task, while *Random Walk* is the easiest.

5.2 Data Augmentation for VLN

Data augmentation using instructions from the Speaker-Follower and EnvDrop models is pervasive in the training of VLN agents (Wang et al., 2019; Huang et al., 2019; Li et al., 2019). In this section we evaluate whether our compatibility model can be used to filter out low quality instructions from the augmented training set to improve VLN performance. We score all of 170k augmented instruction-trajectory pairs from the Speaker-Follower model and rank them in descending order. We then use different fractions of the ranked data to train VLN agents, and compare with agents trained using random samples of the same size. We use a VLN agent model based on Wang et al. (2019) and implemented in VALAN (Lansing et al., 2019), which achieves a success rate (SR) of 45% on the R2R val-unseen split when trained on

the R2R train split and all of the Speaker-Follower augmented instructions.

Figure 2 indicates that instruction-trajectory pairs selected by our compatibility model consistently outperform random training pairs in terms of the performance of the trained VLN agent. This demonstrates the efficacy of our compatibility model for improving VLN data augmentation by identifying high quality instructions.

5.3 Correlation with Human Wayfinders

In this section we evaluate the correlation between the scores given by our instruction-trajectory compatability model and the outcomes from the human wayfinding attempts described in Section 3. Using Kendall's τ to assess rank correlation, we report both system-level and instance-level correlation. The instance-level evaluations assess whether the metric can identify the best *instruction* from two candidates, while the system-level evaluations assess whether a metric can identify the best *model* from two candidates (after averaging over many instruction scores for each model). The results in Table 3 are reported separately over all 3.9k instructions (9 systems comprising the rows of Table 1), and over model-generated instructions only (4 systems comprising the 2.2k instructions generated by the Speaker-Follower and EnvDrop models on R2R val-seen and val-unseen).

Automatic Metrics For comparison we include standard textual evaluation metrics (BLEU, CIDEr, METEOR, ROUGE and SPICE) and two model-based metrics: BERTScore (Zhang et al., 2019), and scores based on the performance of a trained VLN agent attempting to follow the candidate instruction (Agarwal et al., 2019). Note that only the compatibility model and the VLN agent-based scores use the candidate trajectory – the other metrics are calculated by comparing each candidate instruction to the three reference instructions from R2R (and are thus reliant on reference instructions).

To calculate the standard metrics we use the official evaluation code provided with the COCO captions dataset (Chen et al., 2015). For BERTScore, we use a publicly available uncased BERT[4] model with 12 layers and hidden dimension 768, and compute the mean $F1$-score over the three references. For the VLN agent score, we train three VLN agents based on Wang et al. (2019) from different random initializations using the R2R train set.

[4] tfhub.dev/google/bert_uncased_L-12_H-768_A-12/1

			All Instructions (N=3.9k, M=9)			
	Score	Ref	NE ↓	SR ↑	SPL ↑	Quality ↑
System-Level	BLEU-4	✓	(0.00, 0.33)	(-0.22, 0.39)	(-0.22, 0.00)	(0.11, 0.39)
	CIDEr	✓	(0.06, 0.39)	(-0.22, 0.39)	(-0.22, 0.00)	(0.17, 0.39)
	METEOR	✓	(0.11, 0.44)	(-0.39, 0.28)	(-0.39, -0.06)	(0.00, 0.28)
	ROUGE	✓	(0.06, 0.39)	(-0.28, 0.39)	(-0.33, 0.00)	(0.06, 0.39)
	SPICE	✓	(**-0.67**, **-0.28**)	(**-0.06**, **0.61**)	(**0.44**, **0.78**)	(**0.56**, **0.83**)
	BERTScore	✓	(0.06, 0.39)	(-0.22, 0.39)	(-0.22, 0.00)	(0.17, 0.39)
	$SPL_{1\text{-agent}}$		(-0.50, -0.06)	(-0.22, 0.44)	(0.11, 0.56)	(0.00, 0.44)
	$SPL_{3\text{-agents}}$		(-0.22, 0.17)	(-0.33, 0.39)	(0.00, 0.33)	(0.33, 0.61)
	$SDTW_{1\text{-agent}}$		(-0.44, 0.00)	(-0.22, 0.44)	(0.11, 0.50)	(0.00, 0.44)
	$SDTW_{3\text{-agents}}$		(-0.22, 0.17)	(-0.28, 0.33)	(0.00, 0.33)	(0.33, 0.61)
	Compatibility		(-0.17, 0.17)	(-0.17, 0.50)	(0.00, 0.28)	(0.44, 0.72)

			All Instructions (N=3.9k, M=9)			
	Score	Ref	NE ↓	SR ↑	SPL ↑	Quality ↑
Instance-Level	BLEU-4	✓	(0.05, 0.09)	(-0.04, 0.00)	(-0.09, -0.05)	(-0.01, 0.03)
	CIDEr	✓	(0.06, 0.09)	(-0.04, -0.00)	(-0.11, -0.07)	(-0.02, 0.01)
	METEOR	✓	(0.00, 0.04)	(-0.05, -0.02)	(-0.04, 0.00)	(-0.01, 0.02)
	ROUGE	✓	(0.05, 0.08)	(-0.05, -0.01)	(-0.10, -0.06)	(-0.02, 0.02)
	SPICE	✓	(-0.05, -0.02)	(-0.00, 0.04)	(0.03, 0.06)	(0.03, 0.07)
	BERTScore	✓	(-0.04, -0.00)	(0.07, 0.12)	(-0.01, 0.03)	(0.07, 0.11)
	$SPL_{1\text{-agent}}$		(-0.18, -0.14)	(0.15, 0.19)	(0.14, 0.18)	(0.07, 0.11)
	$SPL_{3\text{-agents}}$		(**-0.22**, **-0.18**)	(**0.20**, **0.24**)	(**0.18**, **0.22**)	(0.10, 0.14)
	$SDTW_{1\text{-agent}}$		(-0.18, -0.14)	(0.15, 0.19)	(0.14, 0.18)	(0.08, 0.12)
	$SDTW_{3\text{-agents}}$		(**-0.22**, **-0.19**)	(**0.20**, **0.24**)	(**0.18**, **0.22**)	(0.11, 0.15)
	Compatibility		(**-0.20**, **-0.17**)	(0.13, 0.17)	(**0.17**, **0.20**)	(**0.19**, **0.23**)

			Model-Generated Instructions (N=2.2k, M=4)			
	Score	Ref	NE ↓	SR ↑	SPL ↑	Quality ↑
Instance-Level	BLEU-4	✓	(-0.02, 0.03)	(-0.03, 0.02)	(-0.02, 0.03)	(-0.02, 0.03)
	CIDEr	✓	(-0.02, 0.03)	(-0.03, 0.02)	(-0.02, 0.03)	(-0.02, 0.03)
	METEOR	✓	(-0.02, 0.03)	(-0.03, 0.02)	(-0.02, 0.03)	(-0.02, 0.03)
	ROUGE	✓	(-0.02, 0.03)	(-0.05, 0.00)	(-0.04, 0.01)	(-0.03, 0.02)
	SPICE	✓	(-0.05, -0.00)	(0.00, 0.05)	(0.00, 0.05)	(0.01, 0.06)
	BERTScore	✓	(-0.22, -0.18)	(0.19, 0.24)	(0.18, 0.23)	(0.16, 0.20)
	$SPL_{1\text{-agent}}$		(-0.21, -0.16)	(0.17, 0.23)	(0.16, 0.22)	(0.07, 0.12)
	$SPL_{3\text{-agents}}$		(**-0.26**, **-0.21**)	(**0.21**, **0.27**)	(**0.21**, **0.26**)	(0.09, 0.14)
	$SDTW_{1\text{-agent}}$		(-0.22, -0.16)	(0.17, 0.23)	(0.16, 0.22)	(0.07, 0.13)
	$SDTW_{3\text{-agents}}$		(**-0.26**, **-0.21**)	(**0.22**, **0.27**)	(**0.21**, **0.26**)	(0.10, 0.15)
	Compatibility		(**-0.25**, **-0.20**)	(**0.22**, **0.27**)	(**0.21**, **0.25**)	(**0.18**, **0.23**)

Table 3: Kendall's τ correlation between automated instruction evaluation metrics and human wayfinder evaluations. Ranges are 90% confidence intervals based on bootstrap resampling. **N** refers to the number of instructions and **M** refers to the number of systems. If checked, **Ref** indicates that the metric requires reference instructions for comparison. $SPL_{k\text{-agent(s)}}$ and $SDTW_{k\text{-agent(s)}}$ refers to wayfinding scores averaged over k VLN agents trained from random initialization.

We then employ the trained agents for the wayfinding task and report performance as either the SPL or SDTW similarity between the path taken by the agent and the reference path – using either a single agent or the average score from three agents.

Results Table 3 compares system-level and instance-level correlations for all metrics, both standard and model-based. At the *system-level*, we see no correlation between standard text metrics such as BLEU, ROUGE, METEOR and CIDEr and human wayfinder performance. The exception is SPICE, which shows the desired negative correlation with NE, and positive correlation with SR, SPL (see Figure 3) and Quality. At the system-level, the model-based approaches (BERTScore,

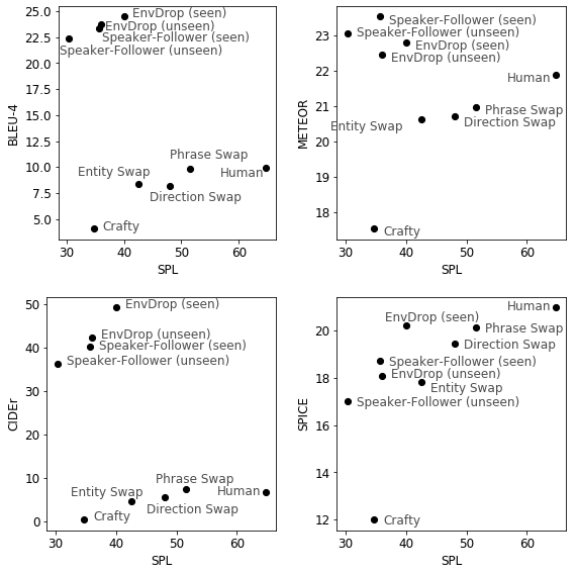

Figure 3: Standard evaluation metrics vs. human wayfinding outcomes (SPL) for 9 navigation instruction generation systems. SPICE is most consistent with human wayfinding outcomes, although no metrics score the Crafty template-based instructions highly.

agent SPL/SDTW and Compatability) also lack the desired correlation and exhibit wide confidence intervals. Here, it is important to point out that the 9 systems under evaluation include a variety of styles (e.g., Crafty's template-based instructions, different annotator pools, adversarial perturbations) which are dissimilar to the R2R data used to train the VLN agents and the compatibility model. Accordingly, the model-based approaches are unable to reliably rank these out-of-domain systems.

At the *instance-level* (when scoring individual instructions) we observe different outcomes. SPICE scores for individual instructions have high variance, and so SPICE does not correlate with wayfinder performance at the instruction level. In contrast, the model-based approaches exhibit the desired correlation, particularly when restricted to the model-generated instructions (Table 3 bottom panel). Our compatibility score shows the strongest correlation among all metrics, performing similarly to an ensemble of three VLN agents.

6 Conclusion

Generating grounded navigation instructions is one of the most promising directions for improving the performance of VLN wayfinding agents, and a challenging and important language generation task in its own right. In this paper, we show that efforts to improve navigation instruction generators have been hindered by a lack of suitable automatic evaluation metrics. With the exception of SPICE, all the standard textual evaluation metrics we evaluated (BLEU, CIDEr, METEOR and ROUGE) are ineffective, and – perhaps as a result – existing instruction generators have substantial headroom for improvement.

To address this problem, we develop an instruction-trajectory compatibility model that outperforms all existing automatic evaluation metrics on instance-level evaluation without needing any reference instructions – making it suitable for use as a reward function in a reinforcement learning setting, as a discriminator in a Generative Adversarial Network (GAN) (Dai et al., 2017), or for filtering instructions in a data augmentation setting.

Progress in natural language generation (NLG) is increasing the demand for evaluation metrics that can accurately evaluate generated text in a variety of domains. Our findings are a timely reminder that textual evaluation metrics should not be trusted in new domains unless they have been comprehensively validated against human judgments. In the case of grounded navigation instructions, for model selection in the presence of reference instructions we recommend using the SPICE metric. In all other scenarios (e.g., selecting individual instructions, or model selection without reference instructions) we recommend using a learned instruction-trajectory compatibility model.

Acknowledgements

We thank the Google Data Compute team, in particular Ashwin Kakarla and Priyanka Rachapally, for their tooling and annotation support for this project.

References

Sanyam Agarwal, Devi Parikh, Dhruv Batra, Peter Anderson, and Stefan Lee. 2019. Visual landmark selection for generating grounded and interpretable navigation instructions. In *CVPR workshop on Deep Learning for Semantic Visual Navigation*.

Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. 2018a. On evaluation of embodied navigation agents. *CoRR*, abs/1807.06757.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: Semantic Propositional Image Caption Evaluation. In *ECCV*.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018b. Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*.

Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *CVPR*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*.

Xinlei Chen, Tsung-Yi Lin Hao Fang, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick. 2015. Microsoft COCO Captions: Data Collection and Evaluation Server. *arXiv preprint arXiv:1504.00325*.

Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. 2017. Towards diverse and natural image descriptions via a conditional gan. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Markus Dräger and Alexander Koller. 2012. Generation of landmark-based navigation instructions from open-source data. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 757–766, Avignon, France. Association for Computational Linguistics.

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. In *NeurIPS*.

Tsu-Jui Fu, Xin Eric Wang, Matthew Peterson, Scott Grafton, Miguel Eckstein, and William Yang Wang. 2019. Counterfactual vision-and-language navigation via adversarial path sampling.

Haoshuo Huang, Vihan Jain, Harsh Mehta, Alexander Ku, Gabriel Magalhaes, Jason Baldridge, and Eugene Ie. 2019. Transferable representation learning in vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. 2019. Effective and general evaluation for instruction conditioned navigation using dynamic time warping. *NeurIPS Visually Grounded Interaction and Language Workshop*.

Vihan Jain, Gabriel Magalhães, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. 2019. Stay on the path: Instruction fidelity in vision-and-language navigation. In *ACL*.

Da-Cheng Juan, Chun-Ta Lu, Zhen Li, Futang Peng, Aleksei Timofeev, Yi-Ting Chen, Yaxi Gao, Tom Duerig, Andrew Tomkins, and Sujith Ravi. 2020. Ultra fine-grained image semantic embedding. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, page 277–285.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. 2020. 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)*.

Larry Lansing, Vihan Jain, Harsh Mehta, Haoshuo Huang, and Eugene Ie. 2019. VALAN: Vision and language agent navigation. *arXiv:1912.03241*.

Xiujun Li, Chunyuan Li, Qiaolin Xia, Yonatan Bisk, Asli Celikyilmaz, Jianfeng Gao, Noah Smith, and Yejin Choi. 2019. Robust navigation with language pretraining and stochastic sampling. In *CVPR*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *CVPR*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2014. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. 2020. Improving vision-and-language navigation with image-text pairs from the web. In *CVPR*.

Vivien Mast and Diedrich Wolter. 2013. A probabilistic framework for object descriptions in indoor route instructions. In *Spatial Information Theory*, pages 185–204, Cham. Springer International Publishing.

Harsh Mehta, Yoav Artzi, Jason Baldridge, Eugene Ie, and Piotr Mirowski. 2020. Retouchdown: Adding touchdown to streetlearn as a shareable resource for language grounding tasks in street view. *arXiv:2001.03671*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

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 *CVPR*.

Kai-Florian Richter and Alexander Klippel. 2005. A model for context-specific route directions. In *Spatial Cognition IV. Reasoning, Action, Interaction*, pages 58–78, Berlin, Heidelberg. Springer Berlin Heidelberg.

Michael Roth and Anette Frank. 2010. Computing EM-based alignments of routes and route directions as a basis for natural language generation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 958–966, Beijing, China. Coling 2010 Organizing Committee.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Hao Tan, Licheng Yu, and Mohit Bansal. 2019. Learning to navigate unseen environments: Back translation with environmental dropout. In *NAACL*.

Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. 2019. Vision-and-dialog navigation. In *CoRL*.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. 2019. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *CVPR*.

Yinfei Yang, Gustavo Hernández Ábrego, Steve Yuan, Mandy Guo, Qinlan Shen, Daniel Cer, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2019. Improving multilingual sentence embedding using bi-directional dual encoder with additive margin softmax. In *IJCAI*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with BERT. In *ICLR*.

A Automated metric scores for all instructions

We provide more details about automated metric scores for all instructions in this section. Table 4 gives automated metrics for each model we consider. Generated instructions from EnvDrop and Speaker-Follower are scored the highest, whereas human instructions are scored poorly and on par with perturbed instructions, and Crafty is the lowest. These results diverge significantly from human wayfinding performance in Section 3, and highlights the inefficacy of these automated text metrics.

	BertScore	BLEU-4	CIDEr	ROUGE	METEOR	SPICE
Val-unseen						
Speaker Fol.	78.9	22.3	36.3	45.6	23.1	17.0
EnvDrop	79.3	23.7	42.2	45.8	22.5	18.1
Val-seen						
Speaker Fol.	79.0	23.3	40.1	46.3	**23.5**	18.7
EnvDrop	**79.5**	**24.5**	**49.3**	**46.7**	22.8	20.2
Crafty	71.5	4.1	0.4	23.3	17.5	12.0
Dir. Swap	74.7	8.2	5.7	31.4	20.7	19.4
Entity Swap	74.0	8.4	4.6	31.4	20.6	17.8
Phrase Swap	74.8	9.8	7.4	31.4	21.0	20.1
Human	74.9	9.9	6.7	33.3	21.9	**21.0**

Table 4: Automated metric scores for all instructions. The BLEU, CIDEr and ROUGE metrics score human instructions poorly compared to the neural net models.

B Crafty Details

We use the data in Matterport3D to build Crafty, a template-based navigation instruction generator that uses a Hidden Markov Model (HMM) to select objects as reference landmarks for wayfinding. Crafty's four main components (*Appraiser*, *Walker*, *Observer* and *Talker*) are described below.

B.1 Appraiser

The Appraiser scores the interestingness of objects based on the Matterport3D scans in the training set. It treats each panorama as a document and the categories corresponding to objects visible from the panorama as words, and then computes a per-category inverse document frequency (IDF) score.

B.2 Walker

The Walker converts a panorama sequence into a motion sequence. Given a path (sequence of connected panoramas) and an initial heading, it calculates the entry heading into each panorama and the exit heading required to transition to the next panorama. For each panorama, all annotated objects that are visible from the location are retrieved. For each object, we obtain properties such as their category and center, which allows the distance and heading from the panorama center to be computed. From these, the Walker creates a sequence of motion tuples, each of which captures the context of the source panorama and the goal panorama, along with the heading to move from source to goal.

B.3 Observer

The Observer selects an object sequence by generating objects from an HMM that is specially constructed for each environment, characterized by:

- *Emissions*: how panoramas relate to objects. This is a probability distribution over panoramas for each object, based on the distance between the object and the panoramas.
- *Transitions*: how looking at one object might shift to another one, based on their relative location, the motion at play, and the Appraiser's assessment of their prominence.

The intuition for using an HMM is that we tend to fixate on a given salient object over several steps as we move (high self-transitions); these tend to be nearby (high emission probability for objects near a panorama's center) and connected to the next salient object (biased object-object transitions). To explain a particular observed panorama sequence (path), we can then infer the optimal object sequence using the Viterbi algorithm.

B.4 Talker

Given a motions sequence from the Walker and corresponding object observations from the Observer, the Talker uses a small set of templates to create English instructions for each step. We decompose this into low-level and high-level templates.

B.4.1 Low-level templates

For single step actions, there are three main things to mention: movement, the fixated object and its relationship to the agent's position.

MOVE. For movement, we simply generate a set of possible commands for each direction type, where the direction types are defined as in the orientation wheel shown in Fig. 4. There are additional direction types for UP and DOWN based on relative pitch (e.g. when the goal panorama is higher or lower than the source).

Given one of these heading types, we generate a set of matching phrases appropriate to each. E.g.

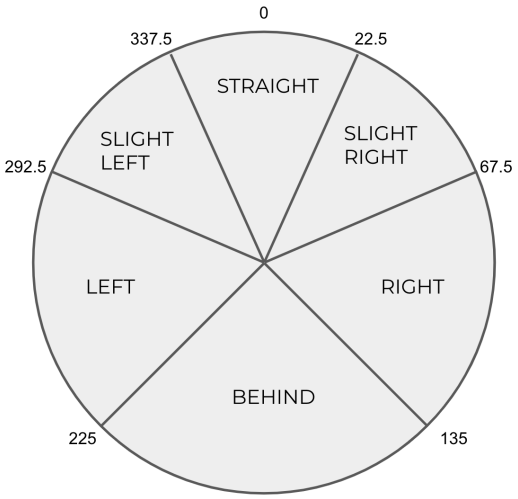

Figure 4: Orientation wheel with directions types. The demarcations are $\frac{\pi}{8}$ (22.5°), $\frac{3\pi}{8}$ (67.5°), etc.

for LEFT and RIGHT, the verbs *face*, *go*, *head*, *make a*, *pivot*, *turn*, and *walk* are combined *left* and *right*, respectively. For moving STRAIGHT, the verbs *continue*, *go*, *head*, *proceed*, *walk*, and *travel* are combined with *straight* and *ahead*. To select an instruction for a given heading, we randomly sample from the list of available phrases. To generate a MOVE command for LEFT, we randomly sample from [*face left*, *go left*, . . . *walk left*].

OBJ. An object's description is its category (e.g. *couch*, *tv*, *window*).

ORIENT. We use the same direction types shown in Figure 4. When an object is STRAIGHT and BEHIND, we use the phrases *ahead of you* or *in front of you* and *behind you* or *in back of you*, respectively. For objects to the LEFT or RIGHT, we use two templates `DIRECTION_PRE DIRECTION` and `DIRECTION DIRECTION_POST`, where `DIRECTION_PRE` is selected from [*to your*, *to the*, *on your*, *on the*] and `DIRECTION_POST` is the phrase *of you*. This produces *to your left*, *on the right*, *right of you*, and so on. For SLIGHT LEFT and SLIGHT RIGHT, one of [*a bit*, *slightly*, *a little*, *just*] is added in front (e.g. *a bit to your left*).

B.4.2 High level templates

Crafty pieces these low-level textual building blocks together to describe actions. In what follows, MOVE, OBJ, and ORIENT indicate the move command, object phrase and orientation phrase, respectively, discussed above.

Single action. We use templates for three situations: start of a path, heading change in a panorama (intra) and moving between panoramas (inter).

- *Start of path*: There are several templates that simply help a wayfinder verify their current position. Ex: *you are near a* `OBJ, ORIENT`.
- *Intra*: These templates include the movement command followed by a verification of the orientation to an object having completed the movement. Ex: `MOVE`. *a* `OBJ` *is* `ORIENT`.
- *Inter* : These templates capture walking from one panorama to another and provide additional object verification. Ex: `MOVE`, *going along to the* `OBJ ORIENT`.

Multi-step actions. We attempt to reduce verbosity by collapsing actions that involve fixation on the same object.

- *Combining actions*: Repeated actions are collapsed; e.g. [STRAIGHT, STRAIGHT, RIGHT, STRAIGHT] becomes [STRAIGHT, RIGHT, STRAIGHT]). These produce a composite move command, e.g. *proceed forward and make a right and go straight*.
- *Describing the object*: To orient with respect to the fixated-upon object, we switch on the direction type between the agent and the object at the last action. Ex: for STRAIGHT, we use *heading toward the* `OBJ` and for SLIGHT LEFT/RIGHT, we use *approaching the* `OBJ ORIENT`.

The final output is the concatenation of the combined move command and the object orientation phrase.

End-of-path instruction templates. The final action is a special situation in that it needs to describing stopping near a salient object. For this, we extract MOVE and OBJ phrases from the last action and use templates such as `MOVE` *and stop by the* `OBJ`.

Full example. Putting it all together, Crafty creates full path instructions such as the following, with relevant high-level templates indicated:

- (START) *there is a lamp when you look a bit to the left. pivot right, so it is in back of you.*
- (INTER) *walk forward, going along to the curtain in front of you.*

- (INTRA) *curve left. you should see a tv ahead of you.*
- (MULTI-ACTION) *go forward and go slightly left and walk straight, passing the curtain to your right.*
- (END-OF-PATH) *continue forward and stop by the couch.*

Crafty's instructions are more verbose than human instructions, but are often easy to follow—provided there are good, visually salient landmarks in the environment to use for orientation.

C Human Rater Performance Over Time

Human raters are excellent at learning and adapting to new problems over time. To understand whether our 37 human raters learn to self-correct the perturbed instructions over time and whether that affects the quality of our human wayfinding results, we investigate rater performance as a function of time using the sequence of examples they evaluate.

Figure 5 shows the average human rater performance for all of the 9 datasets included in Table 1 of Section 3. Due to the binary nature of SR, we use a 50-point bin to average each rater's performance, and then average the results across all raters for each bin. Figure 5 shows that the average rater performance stays flat within the uncertainties and does not show systematic drift over time, indicating no overall self-correction that affects the wayfinding results. For a more granular scrutiny of individual perturbation methods, in particular the perturbed instructions, we plot in Figure 6 the average human rater performance over time for the three methods: *Direction Swap*, *Entity Swap*, and *Phrase Swap*. Despite greater uncertainties due to much fewer data points used for averaging, the overall human performance for each method still does not drift significantly in a systematic manner. These results indicate that our human wayfinding performance results are reliable and robust over time, which can be attributed to shuffling of the examples and that the perturbation methods are blind to human raters.

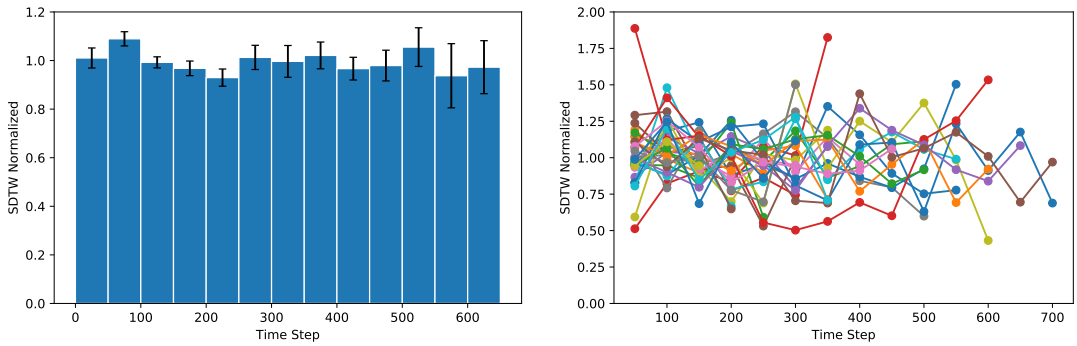

Figure 5: Average human rater performance for all datasets as a function of time using the sequence of examples each rater has evaluated. Each rated example indicates a time step. We normalize the scores of each rater by their mean value over time to remove performance bias of each rater in order to better pick up the trend over time. We average 50 examples to get the mean SDTW for each rater due to the discrete nature of success. *Left*: The mean performance of all rater for each bin. Error bars represent the standard deviation of the mean. *Right*: Individual rater performance over time. Each line represents a single rater. Despite a few outliers, the overall human rater performance is flat and consistent over time, indicating no self-correction or adaptation to the datasets by human raters.

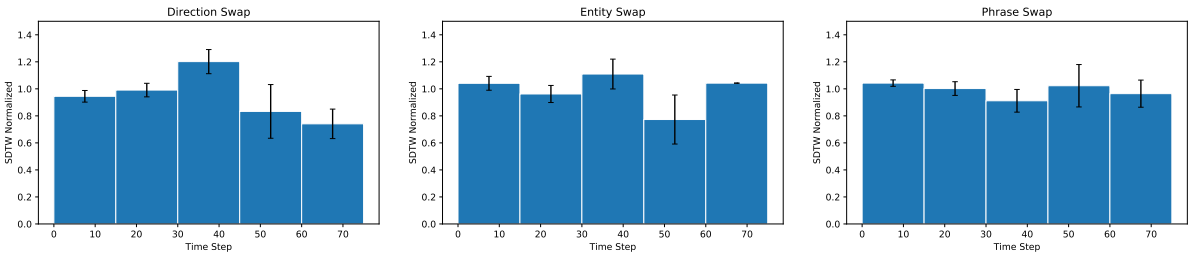

Figure 6: Average human rater performance for the three instruction perturbation methods as a function of time (number of examples), computed in a similar way as in Figure 5. We use a 15-point bin to compute the average for each human rater, and aggregate over all raters to get the mean and its uncertainty. The overall human rater performance stays flat and does not drift significantly over time for instruction perturbations.