

Effective and General Evaluation for Instruction Conditioned Navigation using Dynamic Time Warping

Gabriel Magalhaes Vihan Jain Alexander Ku Eugene Ie Jason Baldridge
Google Research
{gamaga, vihan, alexku, eugeneie, jridge}@google.com

Abstract:

In instruction conditioned navigation, agents interpret natural language and their surroundings to navigate through an environment. Datasets for studying this task typically contain pairs of these instructions and reference trajectories. Yet, most evaluation metrics used thus far fail to properly account for the latter, relying instead on insufficient similarity comparisons. We address fundamental flaws in previously used metrics and show how **Dynamic Time Warping (DTW)**, a long known method of measuring similarity between two time series, can be used for evaluation of navigation agents. For such, we define the normalized Dynamic Time Warping (**nDTW**) metric, that softly penalizes deviations from the reference path, is naturally sensitive to the order of the nodes composing each path, is suited for both continuous and graph-based evaluations, and can be efficiently calculated. Further, we define **SDTW**, which constrains nDTW to only successful paths.¹ We collect **human similarity judgments** for simulated paths and find nDTW correlates better with human rankings than all other metrics. We also demonstrate that **using nDTW as a reward signal for Reinforcement Learning navigation agents improves their performance** on both the Room-to-Room (R2R) and Room-for-Room (R4R) datasets. **The R4R results in particular highlight the superiority of SDTW** over previous success-constrained metrics.

Keywords: Human-Robot Interaction. Grounding Language Instructions.

1 Introduction

Following natural language instructions is essential for flexible and intuitive interaction between humans and embodied agents. Recent advances in machine learning and high availability of processing power has greatly lessened some of the technical barriers for learning these behaviours. In particular, there has been growing interest in the task of Vision-and-Language Navigation (VLN), where agents use language instructions and visual stimuli to navigate in an environment. These can be both virtual—sometimes photo-realistic—[1, 2, 3, 4, 5] or physical environments [6, 7, 8]. Scoring agent behaviors appropriately, however, is not a straightforward matter. For behaviors involving language, scores should be sensitive to both the task and to how language informs the task. In the context of VLN, there are different kinds of instructions, including:

- Goal-oriented instructions [2, 4]: *Go to the bedroom with the striped wallpaper.*
- Path-oriented instructions [3, 5]: *Head upstairs and walk past the piano through an archway directly in front. Turn right when the hallway ends at pictures and table. Wait by the moose antlers hanging on the wall.*
- Path-oriented instructions with tasks [1]: *Go back to the kitchen, put the glass in the sink.*

¹Evaluation code for nDTW and SDTW can be found at <https://github.com/google-research/google-research/tree/master/r4r>

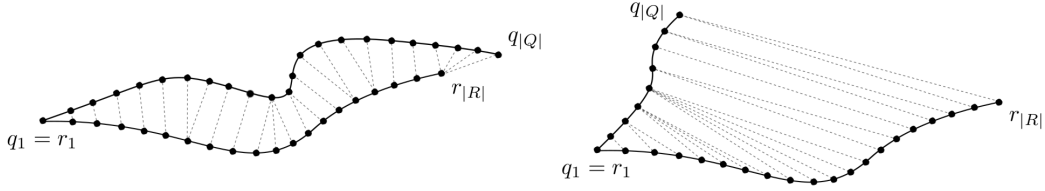


Figure 1: Illustration of two pairs of reference ($R = r_{1..|R|}$) and query ($Q = q_{1..|Q|}$) series (solid), and the optimal warping between them (dashed) when computing DTW.

The most obvious way of evaluating agents is whether they succeed in getting to their goals. Success rate is well suited to goal-oriented instructions, but it breaks down in the other situations. In the short time since its introduction, for instance, the Room-to-Room dataset [3] has seen a continued refinement of metrics that best capture the desired behaviors of its path-oriented instructions. It began with success rate as the primary metric, but this was exploited by agents that did considerable exploration and were less concerned with efficient, effective use of the instructions. This led to the introduction of a metric that weighted success by the optimal path length [9], but recently this was shown to be inadequate for the case where paths involve multiple segments and involve turns [10].

We apply Dynamic Time Warping (DTW) [11] to assess the performance of navigational agents. DTW is a similarity function for time series that has long been known and used in speech processing [12, 13, 14], robotics [15, 16], data mining [17, 18], handwriting recognition [19], gesture recognition [20, 21] and many other areas [22, 23, 24]. DTW identifies an optimal warping, which is the alignment of elements from a reference and a query series such that the cumulative distance between aligned elements is minimized, as illustrated in Figure 1.

Our $nDTW$ metric (normalized Dynamic Time Warping) has many desirable properties for scoring paths with respect to one another.

1. It measures similarity between the entirety of two trajectories, softly penalizing deviations.
2. It naturally captures the importance of the goal by forcing the alignment between the final nodes of the trajectories.
3. It is insensitive to changes in scale and density of nodes.
4. It is sensitive to the order in which nodes compose trajectories, which is especially important in cases where paths double back on themselves.
5. It can be used for continuous path evaluation [25] as well as graph-based evaluation.
6. It can be computed exactly in quadratic time and has an efficient linear time approximation.
7. It is relevant in robotics beyond language instruction tasks. It can be used for any task requiring matching a sequence of actions, such as a robot following a human demonstration of an action sequence, provided a element-wise distance function is chosen.

We demonstrate $nDTW$'s utility by showing that it correlates much better with human judgments of path comparisons than all previous metrics. We also define $SDTW$, a success rate metric weighted by $nDTW$, which is superior to other metrics when goal completion is paramount. Furthermore, we improve performance for both Room-to-Room [3] and Room-for-Room [10] by using $nDTW$ scores as reward signals for Reinforcement Learning (RL) agents in VLN tasks.

2 Evaluation Metrics in Instruction Conditioned Navigation

Table 1 defines existing metrics for instruction conditioned navigation. There are two distinct scenarios where we want to evaluate the performance of agents. The first is the discrete case where the environment is represented by a graph where nodes are possible locations for the agent and edges are navigable direct paths between nodes [3]. The second is a continuous case where the agent can move freely through non-discrete actions in a navigation space [25]. An ideal evaluation metric should be able to gracefully handle both scenarios. All previous metrics fall short in different ways.

Metric	$\uparrow \downarrow$	Definition
Path Length (PL)	-	$\sum_{1 \leq i < Q } d(q_i, q_{i+1})$
Navigation Error (NE)	\downarrow	$d(q_{ Q }, r_{ R })$
Oracle Navigation Error (ONE)	\downarrow	$\min_{q \in Q} d(q, r_{ R })$
Success Rate (SR)	\uparrow	$\mathbb{1}[\text{NE}(R, Q) \leq d_{th}]$
Oracle Success Rate (OSR)	\uparrow	$\mathbb{1}[\text{ONE}(R, Q) \leq d_{th}]$
Average Deviation (AD)	\downarrow	$\sum_{q \in Q} d(q, R) / Q $
Max Deviation (MD)	\downarrow	$\max_{q \in Q} d(q, R)$
Success weighted by PL (SPL)	\uparrow	$\text{SR}(R, Q) \cdot \frac{d(q_1, r_{ R })}{\max\{\text{PL}(Q), d(q_1, r_{ R })\}}$
Success weighted by Edit Distance (SED)	\uparrow	$\text{SR}(R, Q) \cdot \left(1 - \frac{\text{ED}(A_R, A_Q)}{\max\{ A_R , A_Q \}}\right)$
Coverage weighted by Length Score (CLS)	\uparrow	$\text{PC}(R, Q) \cdot \text{LS}(R, Q)$
Normalized Dynamic Time Warping (nDTW)	\uparrow	$\exp\left(-\frac{\min_{W \in \mathcal{W}} \sum_{(i_k, j_k) \in W} d(r_{i_k}, q_{j_k})}{ R \cdot d_{th}}\right)$
Success weighted by nDTW (SDTW)	\uparrow	$\text{SR}(R, Q) \cdot \text{nDTW}(R, Q)$

Table 1: Metrics and optimal directions, which agents should minimize (\downarrow) or maximize (\uparrow).

Let \mathcal{P} be the space of possible paths, where each $P \in \mathcal{P}$ is a sequence of observations $p_{1..|P|}$. An evaluation metric that measures the similarity between two paths is then some function $f : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}$, where $f(Q, R)$ maps a query path Q and a reference path R to a real number. We denote by $d(n, m)$ the distance of the shortest path between two **nodes** n and m , and $d(n, P) = \min_{p \in P} d(n, p)$ the shortest distance between a node and a path. In discrete scenarios, $d(n, m)$ can be exactly computed using Dijkstra’s algorithm [26]. In continuous scenarios, one strategy for computing $d(n, m)$ is to divide the environment into a grid of points so that they are at most some error margin of each other. The distance $d(n, m)$ between all pairs of grid points can be efficiently pre-computed [27], and the distance between any pair of points can then be obtained within some error margin. In environments where there are no obstacles, $d(n, m)$ can be computed in constant time by taking the Euclidean distance between the points. Commonly, a threshold distance d_{th} is defined for measuring success.

Of the existing metrics for assessing performance in instruction conditioned navigation, the majority are not intended to measure fidelity between two paths Q and R : *Path Length* (PL) measures the length of the query path, optimally equal to the length of the reference path; *Navigation Error* (NE) measures the distance between the last nodes of the query and reference paths; *Oracle Navigation Error* (ONE) measures the distance from the last node in the reference path to the query path; *Success Rate* (SR) measures whether the last node in the predicted path is within d_{th} of the last node in the reference path; *Oracle Success Rate* (OSR) measures whether the distance between the last node in the reference path and the query path is within d_{th} ; finally, *Success weighted by Path Length* (SPL) [9] weights SR with a normalized path length. None of these metrics take into account the entirety of the reference path R and thus are less than ideal for measuring similarity between two paths. Because they are only sensitive to the the last node in the reference path, these metrics are tolerant to intermediary deviations. As such, **they mask unwanted and potentially dangerous behaviours in tasks where following the desired action sequence precisely is crucial** [10].

Success weighted by Edit Distance (SED) [5] uses the Levenshtein edit distance $\text{ED}(R, Q)$ between the two action sequences $A_R = ((r_1, r_2), (r_2, r_3), \dots, (r_{|R|-1}, r_{|R|}))$ and $A_Q = ((q_1, q_2), (q_2, q_3), \dots, (q_{|Q|-1}, q_{|Q|}))$. When computing $\text{ED}(R, Q)$, SED does not take into account the distance between path components, but only checks if the actions are a precise match or not. This shortcoming becomes clear in continuous or quasi-continuous scenarios: **an agent that travels extremely close to the reference path—but not exactly on it—is severely penalized by SED.**

Coverage weighted by Length Score (CLS) [10] computes the path coverage $\text{PC}(R, Q) = \frac{1}{|R|} \sum_{r \in R} \exp\left(-\frac{d(r, Q)}{d_{th}}\right)$ and a length score $\text{LS}(R, Q) = \frac{\text{PC}(R, Q) \cdot \text{PL}(R)}{\text{PC}(R, Q) \cdot \text{PL}(R) + |\text{PC}(R, Q) \cdot \text{PL}(R) - \text{PL}(Q)|}$, combining them by multiplication. Although it addresses the major concerns of previous metrics, CLS is not ideal in some scenarios. For instance, because $\text{PC}(R, Q)$ is order-invariant, for a given reference path $R = (a, b, c, a)$, an agent that navigates $Q_1 = (a, c, b, a)$ and one that executes a trajectory $Q_2 = (a, b, c, a)$ both have the same CLS score. For instance, if a instruction such as “Pick

up the newspaper in the front door, leave it in my bedroom and come back” is given, an agent that navigates along the intended path in the reverse order would be incapable of completing its task.

We include two additional simple metrics that capture a single summary measure of the difference between two paths. *Average Deviation* (AD) and *Max Deviation* (MD) measure the average and maximum deviations from points on the query path, with respect to the entire reference path. Although these metrics take into account both paths in their totality and measure similarity to some extent, they are critically flawed by not taking into account the order of the nodes. Despite its simplicity, we show in Section 4 that AD correlates surprisingly well with human judgments and even outperforms CLS (but not nDTW).

3 Dynamic Time Warping

DTW is computed by aligning elements from a reference and a query series, preserving the order in which elements appear in each of them, and forcing the initial and final elements of the query series to be aligned those of the reference series. Formally, let $R = r_{1..|R|}$ and $Q = q_{1..|Q|}$ be the reference and query series, where $r_i \in \mathcal{F}$, $1 \leq i \leq |R|$ and $q_j \in \mathcal{F}$, $1 \leq j \leq |Q|$ for some feature space \mathcal{F} (in the context of navigation, for instance, \mathcal{F} is the space of navigable points in space). Let $\delta : \mathcal{F} \times \mathcal{F} \rightarrow \mathbb{R}_{\geq 0}$ be a distance function mapping pairs of elements in the feature space \mathcal{F} to a real non-negative number. Let $W = w_{1..|W|}$ be a *warping* where $w_k = (i_k, j_k) \in [1 : |R|] \times [1 : |Q|]$ for $1 \leq k \leq |W|$, satisfying the two conditions below:

- (i) **Boundaries:** $w_1 = (1, 1)$ and $w_{|W|} = (|R|, |Q|)$;
- (ii) **Step size:** $w_{k+1} - w_k \in (1, 1), (1, 0), (0, 1)$, for every $1 \leq k < |W|$.

From (ii), a *monotonicity property* can also be derived: it follows that $i_k \leq i_{k+1}$ and $j_k \leq j_{k+1}$, for every $1 \leq k < |W|$, ensuring the *warping preservers ordering*. Finally, let \mathcal{W} be the set of all valid warpings. Intuitively, \mathcal{W} defines a space of non-linear, order-preserving warpings between the two sequences, respecting the alignment between the initial and final elements in them.

DTW then finds the optimal ordered alignment between the two series by minimizing the cumulative **cost of the warping**. Formally:

$$\text{DTW}(R, Q) = \min_{W \in \mathcal{W}} \sum_{(i_k, j_k) \in W} \delta(r_{i_k}, q_{j_k}). \quad (1)$$

3.1 Dynamic programming implementation

A classic way of computing DTW is to define a matrix $\mathbf{C} \in \mathbb{R}^{(|R|+1) \times (|Q|+1)}$ where

$$\mathbf{C}_{i,j} := \text{DTW}(r_{1..i}, q_{1..j}) \quad (2)$$

for $(i, j) \in [0 : |R|] \times [0 : |Q|]$. All elements in this matrix can be computed in $O(|R||Q|)$, using dynamic programming, as shown in Algorithm 1. The key to do so is realizing that $\mathbf{C}_{i,j}$ depends only on $\mathbf{C}_{i-1,j}$, $\mathbf{C}_{i,j-1}$ and $\mathbf{C}_{i-1,j-1}$. Therefore, we can efficiently compute $\text{DTW}(R, Q) = \mathbf{C}_{|R|,|Q|}$ by filling out the slots in matrix \mathbf{C} in an ordered fashion: rows are sequentially computed from 1 to $|R|$ and, in each of them, columns are computed from 1 to $|Q|$. Note that this allows constant time computing of each $\mathbf{C}_{i,j}$, since $\mathbf{C}_{i-1,j}$, $\mathbf{C}_{i,j-1}$ and $\mathbf{C}_{i-1,j-1}$ are previously computed. As initial conditions, $\mathbf{C}_{0,0} = 0$ and $\mathbf{C}_{i,0} = \inf$ and $\mathbf{C}_{0,j} = \inf$, for $1 \leq i \leq |R|$ and $1 \leq j \leq |Q|$.

Lastly, the optimal warping can be computed without increasing time or space complexity. This can be done by using a separate matrix $\mathbf{D} \in \mathbb{R}^{(|R|+1) \times (|Q|+1) \times 2}$ where $\mathbf{D}_{i,j}$ stores the coordinates of the previous cell with the best DTW score. We can traverse to it starting from $\mathbf{D}_{|R|,|Q|}$ while pushing elements into a stack. The optimal warping is given by all elements in this stack.

3.2 FastDTW

Salvador and Chan [28] introduce FastDTW, that approximates DTW in linear time and space complexity. This multilevel approach recursively refines the warping from coarser resolutions using:

Algorithm 1 DTW using dynamic programming (quadratic space and time complexities)

Inputs: The reference series ($R = r_{1..|R|}$); the query series ($Q = q_{1..|Q|}$)

Outputs: The DTW value; the optimal warping

```
procedure DTW( $R, Q$ )
  ▷ Initialize matrices.
   $C = \text{array}(|R| + 1, |Q| + 1)$ 
   $D = \text{array}(|R| + 1, |Q| + 1, 2)$ 
   $C[0][0] \leftarrow 0$ 
  for  $i \leftarrow 1$  to  $|R|$  do
     $C[i][0] \leftarrow \text{inf}$ 
  for  $j \leftarrow 1$  to  $|Q|$  do
     $C[0][j] \leftarrow \text{inf}$ 

  ▷ Compute  $C$  and  $D$ .
  for  $i \leftarrow 1$  to  $|R|$  do
    for  $j \leftarrow 1$  to  $|Q|$  do
       $i^*, j^* = \text{argmin}_{(p,q) \in \{(i-1,j), (i,j-1), (i-1,j-1)\}} C[p][q]$ 
       $C[i][j] \leftarrow \delta(R[i], Q[j]) + C[i^*][j^*]$ 
       $D[i][j] \leftarrow i^*, j^*$ 

  ▷ Compute optimal warping.
  warping_stack = stack()
   $i, j = |R|, |Q|$ 
  while  $(i, j) \neq (0, 0)$  do
    warping_stack.insert( $(i, j)$ )
     $i, j = D[i][j]$ 
  return  $C[|R|][|Q|]$ , warping_stack.pop_all()
```

- (i) **Coarsening**: Reduces the resolution of the time series to fewer data points, aiming to represent the original series as accurately as possible.
- (ii) **Projection**: Finds the optimal warping at the lower resolution and projects the optimal warping back to a higher resolution.
- (iii) **Refinement**: Makes local adjustments to the optimal warping found from the projection.

The algorithm runs recursively decreasing the resolution at each level. Once the resolution is small enough, dynamic programming is executed to compute DTW. At each level, the projected path from the lower resolution is used as a heuristic for finding the optimal warping. Each point in the lower resolution warping maps to a series of points in the higher resolution, which are used for the refinement step. All points within a *search radius* parameter from the projected warping are also searched, and a newly derived warping is found in linear time.

This algorithm can be implemented in linear time and memory complexities. We refer to Salvador and Chan [28] for a proof and pseudo-code. While FastDTW is not guaranteed to find the optimal warping, it often finds paths that are close to optimal. In scenarios where long paths are common, the computational efficiency of FastDTW affords the opportunity to apply DTW as an evaluation function as well as a reward signal for reinforcement learning agents.

3.3 Dynamic Time Warping for Navigation

DTW can be straightforwardly adapted to the context of navigation by using the shortest distance along the graph from node r_i and q_j as the cost function ($\delta(r_i, q_j) = d(r_i, q_j)$). In continuous scenarios where obstruction is not an issue, the Euclidean distance between the coordinates of any two points in space can be used as a cost function. However in continuous scenarios where obstruction is an issue, one can pre-compute pairwise distances from fixed grid points, and approximate the distance at runtime by finding the closest grid points to r_i and q_j .

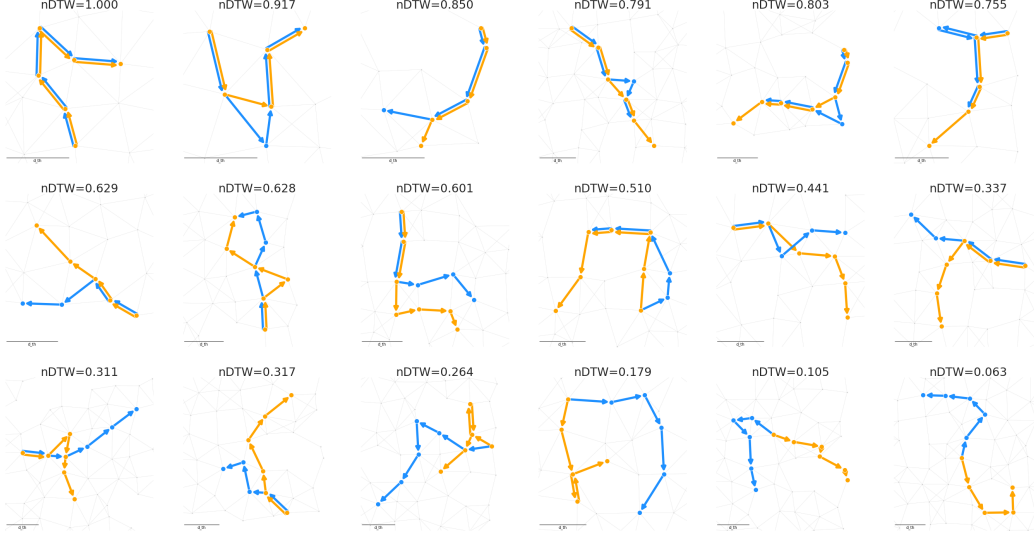


Figure 2: Examples of random reference (blue) and query (orange) paths, sorted by nDTW values.

One important design consideration is to ensure that the similarity function is **invariant to scale**, thus making scenarios like indoor [3] and outdoor [5, 29, 30] navigation more comparable. Further, an ideal metric should be **invariant to the density of nodes** along the trajectories: for instance, in the continuous scenarios, it would be undesirable if the metric changed its value if the sampling rate of the agent changed. Since DTW is in its essence a sum comprised by at least $|R|$ distance terms, **normalizing it by a factor of $\frac{1}{|R| \cdot d_{th}}$ alleviates both of these issues**. Finally, **to aid interpretability, we take the negative exponential of its normalized value**, resulting in a score bounded between 0 and 1, yielding higher numbers for better performance. In summary, *normalized Dynamic Time Warping* (nDTW) is composed by these operations sequentially applied to DTW, as shown in Eq. 3. Figure 2 illustrates multiple pairs of reference (blue) and query (orange) paths, accompanied (and sorted) by their nDTW values.

$$\text{nDTW}(R, Q) = \exp \left(- \frac{\text{DTW}(R, Q)}{|R| \cdot d_{th}} \right) = \exp \left(- \frac{\min_{W \in \mathcal{W}} \sum_{(i_k, j_k) \in W} d(r_{i_k}, q_{j_k})}{|R| \cdot d_{th}} \right) \quad (3)$$

Since it is directly derived from DTW, nDTW can be exactly computed in quadratic time and space complexity as described in Section 3.1 and approximately computed in linear time and space complexity, as described in Section 3.2. **To adapt FastDTW for discrete environments, we modify the *coarsening* step to accommodate the fact that nodes as discrete structures cannot be averaged**. Instead, a random node in each segment can be chosen.

Finally, we notice that, in some tasks, whether or not the trajectory ends close to the goal is key for evaluating performance. This has given rise to multiple “Success weighted by X” metrics, such as SPL and SED. These metrics harshly penalize—by giving a zero score—any path that does not end within the success threshold d_{th} . For such scenarios, we analogously define **Success weighted by normalized Dynamic Time Warping (SDTW)**.

$$\text{SDTW}(R, Q) = \text{SR}(R, Q) \cdot \text{nDTW}(R, Q) \quad (4)$$

As we show in the next section using correlation with human judgments, nDTW is superior to path length, edit distance and CLS for quantifying this latter quantity—and thus SDTW should be preferred to SPL, SED and other success weighted measures. Further, there may be different and multiple success criteria for any given task in addition to goal-oriented SR. For example, **success could mean picking up an item midway through the path and depositing it near**, but not at the end, or selecting a point at the end, as in the spatial description resolution task incorporated into the Touchdown dataset [5]. **Success is then some soft or hard measure of performance on these actions and SDTW captures well both this success criteria and the fidelity between the agent’s movement and the reference path**.

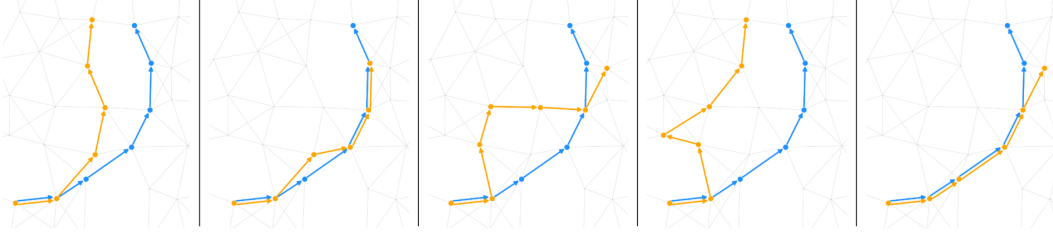


Figure 3: Example comparison set with one reference path (blue) and five query paths (orange) for human raters to rank according to their preferences.

Scenario	PL	NE	ONE	SR	OSR	AD	MD	SPL	SED	CLS	nDTW	SDTW
UC	.15	.70	.60	.25	.30	.75	.69	.26	.30	.57	.76	.25
	$\pm .53$	$\pm .37$	$\pm .43$	$\pm .53$	$\pm .51$	$\pm .33$	$\pm .33$	$\pm .51$	$\pm .49$	$\pm .49$	$\pm .33$	$\pm .54$
SC	.01	.20	.15	n/a	n/a	.74	.68	.12	.16	.73	.81	.81
	$\pm .53$	$\pm .54$	$\pm .55$	n/a	n/a	$\pm .31$	$\pm .37$	$\pm .53$	$\pm .54$	$\pm .31$	$\pm .28$	$\pm .28$

Table 2: Average and standard deviation of Spearman’s rank correlations of navigation metrics and human rankings, on **unconstrained (UC)** and success-constrained (SC) scenarios. See Table 1 for definition of each metric.

4 Evaluation

To assess the utility of nDTW as a measure of similarity between two paths, we compare its correlation with human judgments for simulated paths in the context of other standard metrics. Further, we demonstrate that it is advantageous to use it as a reward signal for RL agents on the Room-to-Room (R2R) task [3] and its Room-for-Room (R4R) extension [10].

4.1 Human evaluations

To better understand how evaluation metrics behave, we turn to human judgment. As illustrated in Figure 3, we give human raters a series of questionnaires each containing a set of five reference (shown in blue) and query (shown in orange) path pairs. In each set, we keep the reference path fixed and instruct raters to ask themselves the following question when ranking the path pairs: “If I instructed a robot to take the blue path, which of these orange paths would I prefer it to take?”

The environment and paths are randomly generated. The environment consists of 15×15 nodes, forming an approximate grid. Each node n_{ij} , $1 \leq i, j \leq 15$ has coordinates (x_{ij}, y_{ij}) , where $x_{ij} \sim U(i - \zeta, i + \zeta)$ and $y_{ij} \sim U(j - \zeta, j + \zeta)$ are independently drawn according to a parameter ζ set to 0.3. For every pair of nodes $n_{i_1 j_1}$ and $n_{i_2 j_2}$, we take the Euclidean distance between their coordinates $\|n_{i_1 j_1} n_{i_2 j_2}\|$ and connect them with an edge if and only if $\|n_{i_1 j_1} n_{i_2 j_2}\| \leq 1.4$. Each path is generated according to a random procedure: first, a random node is drawn; then, a node two or three edges away is chosen, and this step is repeated. The final path is the shortest path connecting adjacent nodes from this procedure. Figure 3 illustrates some of these paths. As in Anderson et al. [3], we set the success threshold to be 1.33 times the average edge length in the environment.

Since some metrics only give a score greater than zero if the success criteria is met, we study two scenarios, *unconstrained* (UC) and *success-constrained* (SC). In the first, query paths are randomly generated without constraints, while in the second, exemplified in Figure 3, only query paths that meet the success criteria are considered.

We collect annotations on 2525 samples (505 sets of 5 query and reference pairs) from 9 human raters, split between UC (1325 samples) and SC (1200 samples). Each set is ranked according to the metrics in Table 1. Once scores are computed, we compare the rankings of each metric with the ones generated by humans, and calculate **the average Spearman’s rank correlation** [31]. The correlation scores are shown in Table 2, for all metrics presented in Table 1.

The AD and MD numbers show that such simple measures are often better than more complex ones. Nonetheless, nDTW still beats both of them handily across both UC and SC. Furthermore, the lower

Agent	R2R						R4R					
	SR	SPL	SED	CLS	nDTW	SDTW	SR	SPL	SED	CLS	nDTW	SDTW
random	5.1	3.3	5.8	29.0	27.9	3.6	13.7	2.2	16.5	22.3	18.5	4.1
goal-oriented	43.7	38.4	31.9	53.5	54.4	36.1	28.7	15.0	9.6	33.4	26.9	11.4
fidelity-oriented	44.4	41.4	33.9	57.5	58.3	38.3	28.5	21.4	9.4	35.4	30.4	12.6

Table 3: Evaluation metrics as percentages on R2R and R4R Validation Unseen sets for agents with different reward functions. In all metrics, higher means better.

standard deviation for nDTW compared to all others shows that it more consistently agrees with human rankings across all samples.

4.2 Evaluation on VLN Tasks

We demonstrate a practical application of nDTW, by using it as a reward signal for agents in the Matterport3D environment [32], on both the R2R [3] and R4R datasets [10]. Unlike R2R, which contains only direct-to-goal reference paths, **R4R contains more convoluted paths that might even come back to their starting point.** In the latter scenario, the overly simplistic nature of metrics that use only the last node in the reference path to evaluate performance becomes more evident.

We follow the experimental settings of Jain et al. [10], and train agents using our own implementation. As a baseline, we follow Jain et al. [10] for evaluating random agents, by sampling the number of steps from the distribution of those of reference paths in the datasets. Each step is taken by uniformly sampling between possible neighbors. We report the metrics by averaging them across 1 million of these random trajectories.

Our *goal-oriented* agent receives at each transition $q_i \rightarrow q_{i+1}$ a reward equal to how much closer it got to its final goal— $d(q_i, r_{|R|}) - d(q_{i+1}, r_{|R|})$. At the end of each episode, the agent receives a completion reward of +1 if it succeeded ($d(q_{|Q|}, r_{|R|}) \leq d_{th}$, where $d_{th} = 3m$ for the Matterport3D environment) and -1 otherwise. Although this is equivalent of the *goal-oriented* reward of Jain et al. [10], we obtained performance numbers generally higher than those in their work, due to differences in hyper-parameters and implementation details. Our *fidelity-oriented* agent uses nDTW, receiving at each transition a reward proportional to the gain $nDTW(q_{1..i+1}, R) - nDTW(q_{1..i}, R)$. At the end of each episode, the agent receives a non-zero reward only if the success criteria is met, equal to a linear function of its navigation error: $1 - d(q_{|Q|}, r_{|R|})/d_{th}$.

The metrics on the random, goal-oriented and fidelity-oriented agents are shown in Table 3. Compared to a goal-oriented reward strategy, taking advantage of nDTW as a reward signal leads to comparable or better performance as measured by previous metrics, and strictly better performance as measured by both nDTW and SDTW. nDTW shows better differentiation compared to CLS on R4R between goal and fidelity oriented agents. SED scores random paths more highly than those of trained agents, and neither SR nor SED differentiate between goal and fidelity orientation. SPL appears to do so (15.0 vs 21.4), but this is only due to the fact that the **fidelity-oriented agent produces paths that have more similar length to the reference paths rather than fidelity to them.** As such, SDTW provides the clearest signal for indicating both success and fidelity.

5 Conclusion

In this work, we adapt DTW to the context of instruction conditioned navigation to introduce a metric that does not suffer from shortcomings of previous evaluation metrics. The many desirable properties of our proposed metric for evaluating path similarity, nDTW, are reflected both qualitatively in human evaluations—which prefer nDTW over other metrics—and practically in VLN agents—that see performance improvements when using nDTW as a reward signal. For assessing performance of instruction conditioned navigational agents, our proposed SDTW captures well not only the success criteria of the task, but also the similarity between the intended and observed trajectory. While multiple measures (especially path length and navigation error) are useful for understanding different aspects of agent behavior, we hope the community will adopt SDTW as a single summary measure for future work, especially for leaderboard rankings.

References

- [1] D. Misra, A. Bennett, V. Blukis, E. Niklasson, M. Shatkhin, and Y. Artzi. Mapping instructions to actions in 3D environments with visual goal prediction. In *Proc. of EMNLP 2018*, pages 2667–2678, 2018.
- [2] J. Fu, A. Korattikara, S. Levine, and S. Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=r1lq1hRqYQ>.
- [3] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.
- [4] Y. Qi, Q. Wu, P. Anderson, M. Liu, C. Shen, and A. van den Hengel. RERERE: remote embodied referring expressions in real indoor environments. *CoRR*, abs/1904.10151, 2019.
- [5] H. Chen, A. Suhr, D. Misra, and Y. Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Conference on Computer Vision and Pattern Recognition*, 2019.
- [6] T. Kyriacou, G. Bugmann, and S. Lauria. Vision-based urban navigation procedures for verbally instructed robots. *Robotics and Autonomous Systems*, 51(1):69–80, 2005.
- [7] J. Thomason, S. Zhang, R. J. Mooney, and P. Stone. Learning to interpret natural language commands through human-robot dialog. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.
- [8] E. C. Williams, N. Gopalan, M. Rhee, and S. Tellex. Learning to parse natural language to grounded reward functions with weak supervision. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 1–7, 2018. doi:10.1109/ICRA.2018.8460937. URL <https://doi.org/10.1109/ICRA.2018.8460937>.
- [9] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.
- [10] V. Jain, G. Magalhaes, A. Ku, A. Vaswani, E. Ie, and J. Baldridge. Stay on the path: Instruction fidelity in vision-and-language navigation. *arXiv preprint arXiv:1905.12255*, 2019.
- [11] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.
- [12] C. Myers, L. Rabiner, and A. Rosenberg. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(6):623–635, 1980.
- [13] H. Sakoe, S. Chiba, A. Waibel, and K. Lee. Dynamic programming algorithm optimization for spoken word recognition. *Readings in speech recognition*, 159:224, 1990.
- [14] L. Muda, M. Begam, and I. Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *arXiv preprint arXiv:1003.4083*, 2010.
- [15] M. D. Schmill, T. Oates, and P. R. Cohen. Learned models for continuous planning. In *AIS-TATS*, 1999.
- [16] A. Vakanski, I. Mantegh, A. Irish, and F. Janabi-Sharifi. Trajectory learning for robot programming by demonstration using hidden markov model and dynamic time warping. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1039–1052, 2012.

- [17] E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 285–289. ACM, 2000.
- [18] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 262–270. ACM, 2012.
- [19] T. M. Rath and R. Manmatha. Word image matching using dynamic time warping. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–II. IEEE, 2003.
- [20] G. A. Ten Holt, M. J. Reinders, and E. Hendriks. Multi-dimensional dynamic time warping for gesture recognition. In *Thirteenth annual conference of the Advanced School for Computing and Imaging*, volume 300, page 1, 2007.
- [21] A. Akl and S. Valaee. Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, & compressive sensing. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2270–2273. IEEE, 2010.
- [22] B. Legrand, C. Chang, S. Ong, S.-Y. Neo, and N. Palanisamy. Chromosome classification using dynamic time warping. *Pattern Recognition Letters*, 29(3):215–222, 2008.
- [23] U. Rebbapragada, P. Protopapas, C. E. Brodley, and C. Alcock. Finding anomalous periodic time series. *Machine learning*, 74(3):281–313, 2009.
- [24] E. Keogh, L. Wei, X. Xi, M. Vlachos, S.-H. Lee, and P. Protopapas. Supporting exact indexing of arbitrarily rotated shapes and periodic time series under euclidean and warping distance measures. *The VLDB journal*, 18(3):611–630, 2009.
- [25] V. Blukis, D. Misra, R. A. Knepper, and Y. Artzi. Mapping navigation instructions to continuous control actions with position visitation prediction. In *Proceedings of the Conference on Robot Learning*, 2018.
- [26] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.
- [27] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.
- [28] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.
- [29] V. Cirik, Y. Zhang, and J. Baldridge. Following formulaic map instructions in a street simulation environment. *NeurIPS Visually Grounded Interaction and Language Workshop*, 2018.
- [30] P. Mirowski, A. Banki-Horvath, K. Anderson, D. Teplyashin, K. M. Hermann, M. Malinowski, M. K. Grimes, K. Simonyan, K. Kavukcuoglu, A. Zisserman, and R. Hadsell. The streetlearn environment and dataset. *CoRR*, abs/1903.01292, 2019. URL <http://arxiv.org/abs/1903.01292>.
- [31] C. Spearman. The proof and measurement of association between two things. *American journal of Psychology*, 15(1):72–101, 1904.
- [32] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.