General Evaluation for Instruction Conditioned Navigation using Dynamic Time Warping

Gabriel Ilharco[†][*]   **Vihan Jain**[‡]   **Alexander Ku**[‡]   **Eugene Ie**[‡]   **Jason Baldridge**[‡]
[†]Paul G. Allen School of Computer Science & Engineering, University of Washington
[‡]Google Research
gamaga@cs.washington.edu, {vihanjain,alexku,eugeneie,jridge}@google.com

Abstract

In instruction conditioned navigation, agents interpret natural language and their surroundings to navigate in an environment. Datasets for such tasks typically contain pairs of these instructions and reference trajectories, but current popular evaluation metrics fail to properly account for the fidelity of agents to the those trajectories. To address this, we introduce the normalized Dynamic Time Warping (nDTW) metric. nDTW 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 episodes and effectively captures both success and fidelity. We collect human similarity judgments for simulated paths and find our DTW metrics correlates better with human rankings than all other metrics. We also show that using nDTW as a reward signal for agents using reinforcement learning improves performance on both the Room-to-Room and Room-for-Room datasets.

1   Introduction

Following natural language instructions is essential for flexible and intuitive interaction between humans and embodied agents [7]. 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 a virtual—sometimes photo-realistic—[18, 11, 3, 21, 8] or physical environment [15, 29, 32]. For these language conditioned behaviors, evaluation metrics should be sensitive to both the task itself and to how the instructions informs the task.

The most obvious way of evaluating agents in goal-oriented tasks is whether they reach their goals (denoted as *Success Rate* (SR)). However, this measure has its drawbacks when applied to path-oriented tasks [12]. *Success weighted by Path Length* (SPL), introduced by Anderson et al. [2], is an evaluation metric that rewards agents that reached their goals and do so efficiently with respect to the length of their trajectory. Since then, new metrics like *Success weighted by Edit Distance* (SED) [8] and *Coverage weighted by Length Score* (CLS) [12] have been proposed that additionally take into account intermediary states along the agent's trajectory. The analysis performed in Jain et al. [12] (as well as in Appendix of this work) reveal several limitations of these metrics resulting in a gap in the evaluation of such agents.

Dynamic Time Warping (DTW) [4] is a similarity function for time-series which is long used in speech processing [20, 25, 19], robotics [27, 31], data mining [14, 22], handwriting recognition [23],
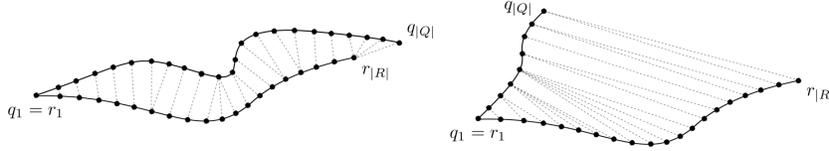

Figure 1: Illustration of two pairs of reference ($R = r_{1..|R|}$) and query ($Q = q_{1..|Q|}$) series (solid), and the optimal warping between them (dashed) when computing DTW.

gesture recognition [28, 1] and more [16, 24, 13]. DTW identifies an optimal warping—an 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. In this work, we adapt the DTW measure for evaluating instruction conditioned navigation agents, demonstrating its strengths against previously proposed metrics and further showing that it can be used as a reward signal to improve training.

2 Dynamic Time Warping for Navigation

DTW is computed by aligning elements of a reference R and a query Q series while preserving their order. If the elements of the two series belong to some feature space $\mathcal{F}$ (in the context of navigation, $\mathcal{F}$ is the space of navigable points in space), then DTW finds the optimal ordered alignment between the two series by minimizing the cumulative cost:

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

where $\delta : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_{\geq 0}$ is some distance function mapping pairs of elements from the two series to a real non-negative number and $W = w_{1..|W|}$ is a *warping* with $w_k = (i_k, j_k) \in [1 : |R|] \times [1 : |Q|]$, respecting the step-size ($w_{k+1} - w_k \in (1, 1), (1, 0), (0, 1)$) and boundaries ($w_1 = (1, 1)$ and $w_{|W|} = (|R|, |Q|)$) conditions.

DTW can be adapted to the context of navigation in discrete environments by using the shortest distance along the environment graph from node r_i and q_j as the cost function. For some applications, the cost function can be the Euclidean distance between the coordinates of any two points in space. In general continuous environments where obstruction can be 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.

The ideal metric for evaluating navigation agents must not be sensitive to the scale and density of nodes along the agent trajectories so that the metric is comparable between different applications (e.g., indoor [3] and outdoor [8, 9, 17]). We adapt DTW—a sum of at least $|R|$ distance terms— by normalizing it by a factor of $\frac{1}{|R| \cdot d_{th}}$, where d_{th} is a sampling rate invariant threshold distance defined for measuring success. Further, to aid interpretability, we take the negative exponential of this 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. 2.

$$\text{nDTW}(R, Q) = \exp\left(-\frac{\text{DTW}(R, Q)}{|R| \cdot d_{th}}\right) = \exp\left(-\frac{\min_{W \in \mathcal{W}} \sum_{(i_k, j_k) \in W} d(r_{i_k}, q_{j_k})}{|R| \cdot d_{th}}\right) \tag{2}$$

Since it is directly derived from DTW, nDTW can be exactly computed in quadratic time and space complexity and approximately computed in linear time and space complexity, as described in the Appendix. In algorithms such as FastDTW [26] where interpolation is required, a simple adaptation—sampling nodes—can be made for discrete environments.

nDTW has many desirable properties for scoring agent paths. It measures similarity between the *entirety* of two trajectories, softly penalizing deviations. It naturally captures the importance of the goal by forcing the alignment between the final nodes of the trajectories. It is insensitive to changes in

Figure 2: Example comparison set with one reference path (blue) and five query paths (orange).

	UC (*nDTW vs*)						**SC** (*SDTW vs*)			
	PL	NE	ONE	CLS	AD	MD	SR	OSR	SPL	SED
+/-	242/17	254/9	255/9	162/46	254/12	253/12	219/16	220/14	219/17	213/26
sign test	4.1e-52	2.0e-63	1.0e-63	2.4e-16	6.9e-60	6.9e-60	9.6e-47	8.8e-49	6.7e-46	1.1e-37

Table 1: Binomial tests on how different metrics compare in correlation with human judgments. The sign test uses n=sum of positives and negatives; k=number of positives; p=0.5.

scale and density of nodes, but sensitive to the order in which nodes compose trajectories.. It can be used for continuous path evaluation [5] as well as graph-based evaluation. The optimal warping can be computed exactly in quadratic time and approximately in linear time. Finally, it is more generally applicable to any task requiring matching a sequence of actions provided an element-wise distance function is available.

Due to the popularity of metrics like SPL [2] and SED [8] that stress the importance of reaching the goal in navigation tasks [3, 8], we analogously define *Success weighted by normalized Dynamic Time Warping* (SDTW), given by $\text{SDTW}(R, Q) = \text{SR}(R, Q) \cdot \text{nDTW}(R, Q)$, where $\text{SR}(R, Q)$ is one if the episode was successful and zero otherwise, commonly defined by the threshold distance d_{th}.

3 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 [12].

3.1 Human evaluations

As illustrated in Figure 2, 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 rank the path pairs in response to this question: *"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} 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. As in Anderson et al. [3], we set the success threshold to be 1.33 times the average edge length in the environment.

We conduct two studies: *unconstrained* (UC) where we compare nDTW with popular continuous metrics like path length (PL), navigation error (NE), oracle navigation error (ONE), CLS [12], average deviation (AD), maximum deviation (MD); and *success-constrained* (SC) where we compare SDTW with metrics that that are non-zero only if a success criteria is met—namely, success rate (SR), oracle success rate (OSR), SPL [2], and SED [8]. These metrics are discussed in detail in the Appendix. 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).

| | R2R | | | | | | R4R | | | | | |
Agent	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 2: Evaluation metrics as percentages on R2R and R4R Validation Unseen sets for agents with different reward functions. In all metrics, higher means better.

To analyze the collected annotations, we first assign nDTW in UC study (similarly SDTW in SC study) a positive/negative sign depending if it has higher/lower correlation than competing metric for a given human ranking of query paths with respect to a reference path, and then compare across all reference paths (discarding ties) using a sign test. Table 1 depicts the results of the binomial test for the null hypothesis. Both nDTW and SDTW correlate substantially better with human orderings, compared to the competing metrics in their respective categories.

3.2 Evaluation on VLN Tasks

We demonstrate a practical application of nDTW, by using it as a reward signal for agents in the Matterport3D environment [6], on both the R2R [3] and R4R datasets [12]. 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 previous metrics clearer. We follow the experimental settings of Jain et al. [12], and train agents using our own implementation. A random baseline as in [12] is computed by sampling the step count from the distribution of reference path lengths in the datasets. Each step is taken by uniformly sampling between possible neighbors and scores are averaged over one 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 the goal g: $d(q_i, g) - d(q_{i+1}, g)$. At the end of the episode q_f, the agent receives a completion reward of +1 if it was deemed successful (i.e., it stopped within an acceptable distance from goal) and -1 otherwise. Our *fidelity-oriented* agent receives at each transition a reward proportional to the gain in nDTW score with respect to the reference path R: $\text{nDTW}(q_{1..i+1}, R) - \text{nDTW}(q_{1..i}, R)$. At the end of the episode, the agent receives a non-zero reward that is a linear function of its navigation error: $1 - d(q_f, g)/d_{th}$ only if it was successful ($d_{th} = 3m$ in Matterport3D environment).

The metrics for the random, goal-oriented and fidelity-oriented agents are shown in Table 2. Compared to a goal-oriented reward strategy, taking advantage of nDTW as a reward signal not only results in better performance on nDTW and SDTW metrics but also better performance on prior metrics like CLS and SPL. 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.

4 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 and leaderboard rankings.

References

[1] Ahmad Akl and Shahrokh 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.

[2] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jagannath Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. On evaluation of Embodied Navigation Agents. *arXiv preprint arXiv:1807.06757*, 2018.

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

[4] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[5] Valts Blukis, Dipendra Misra, Ross A. Knepper, and Yoav Artzi. Mapping navigation instructions to continuous control actions with position visitation prediction. In *Proceedings of the Conference on Robot Learning*, 2018.

[6] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[7] David L. Chen and Raymond J. Mooney. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, pages 859–865. AAAI Press, 2011.

[8] Howard Chen, Alane Suhr, Dipendra Misra, and Yoav Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Conference on Computer Vision and Pattern Recognition*, 2019.

[9] Volkan Cirik, Yuan Zhang, and Jason Baldridge. Following formulaic map instructions in a street simulation environment. *NeurIPS Visually Grounded Interaction and Language Workshop*, 2018.

[10] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[11] Justin Fu, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. In *International Conference on Learning Representations*, 2019.

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

[13] Eamonn Keogh, Li Wei, Xiaopeng Xi, Michail Vlachos, Sang-Hee Lee, and Pavlos 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.

[14] Eamonn J Keogh and Michael 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.

[15] Theocharis Kyriacou, Guido Bugmann, and Stanislao Lauria. Vision-based urban navigation procedures for verbally instructed robots. *Robotics and Autonomous Systems*, 51(1):69–80, 2005.

[16] Benoit Legrand, CS Chang, SH Ong, Soek-Ying Neo, and Nallasivam Palanisamy. Chromosome classification using dynamic time warping. *Pattern Recognition Letters*, 29(3):215–222, 2008.

[17] Piotr Mirowski, Andras Banki-Horvath, Keith Anderson, Denis Teplyashin, Karl Moritz Hermann, Mateusz Malinowski, Matthew Koichi Grimes, Karen Simonyan, Koray Kavukcuoglu, Andrew Zisserman, and Raia Hadsell. The streetlearn environment and dataset. *CoRR*, abs/1903.01292, 2019. URL http://arxiv.org/abs/1903.01292.

[18] Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. Mapping instructions to actions in 3D environments with visual goal prediction. In *Proc. of EMNLP 2018*, pages 2667–2678, 2018.

[19] Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *arXiv preprint arXiv:1003.4083*, 2010.

[20] Cory Myers, Lawrence Rabiner, and Aaron 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.

[21] Yuankai Qi, Qi Wu, Peter Anderson, Marco Liu, Chunhua Shen, and Anton van den Hengel. RERERE: remote embodied referring expressions in real indoor environments. *CoRR*, abs/1904.10151, 2019.

[22] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn 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.

[23] Toni M Rath and Raghavan 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.

[24] Umaa Rebbapragada, Pavlos Protopapas, Carla E Brodley, and Charles Alcock. Finding anomalous periodic time series. *Machine learning*, 74(3):281–313, 2009.

[25] Hiroaki Sakoe, Seibi Chiba, A Waibel, and KF Lee. Dynamic programming algorithm optimization for spoken word recognition. *Readings in speech recognition*, 159:224, 1990.

[26] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[27] Matthew D Schmill, Tim Oates, and Paul R Cohen. Learned models for continuous planning. In *AISTATS*, 1999.

[28] Gineke A Ten Holt, Marcel JT Reinders, and EA 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.

[29] Jesse Thomason, Shiqi Zhang, Raymond J Mooney, and Peter Stone. Learning to interpret natural language commands through human-robot dialog. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[30] John N Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

[31] Aleksandar Vakanski, Iraj Mantegh, Andrew Irish, and Farrokh 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.

[32] Edward C. Williams, Nakul Gopalan, Mina Rhee, and Stefanie 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.

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

Table 3: Metrics and optimal directions, which agents should minimize (↓) or maximize (↑).

Appendix

A Evaluation metrics in instruction conditioned navigation

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 [10]. 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 [30], 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. Table 3 defines existing and proposed metrics for instruction conditioned navigation. All previously proposed metrics fall short in different ways.

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) [2] 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 [12].

Success weighted by Edit Distance (SED) [8] uses the Levenshtein edit distance $\text{ED}(R, Q)$ between the two action sequences $A_R = ((r_1, r_2), (r_2, r_3), ..., (r_{|R|-1}, r_{|R|}))$ and $A_Q = ((q_1, q_2), (q_2, q_3), ..., (q_{|Q|-1}, q_{|Q|}))$. 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) [12] 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. If an 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.

B Implementing of proposed metrics

Algorithm 1 nDTW

Inputs: $R = r_{1..|R|}$ (reference path); $Q = q_{1..|Q|}$ (query path); d_{th} (success distance threshold).
Outputs: The nDTW score.
  procedure NDTW(R, Q, d_{th})
    $\mathbf{C}$ = array([$|R| + 1$, $|Q| + 1$], default=inf)
    $\mathbf{C}[0][0] \leftarrow 0$

    for $i \leftarrow 1$ to $|R|$ **do**
      for $j \leftarrow 1$ to $|Q|$ **do**
        $\mathbf{C}[i][j] \leftarrow \delta(R[i], Q[j]) + \min_{(p,q) \in \{(i-1,j),(i,j-1),(i-1,j-1)\}} \mathbf{C}[p][q]$

    return math.exp(-$\mathbf{C}[|R|][|Q|]$ / ($|R| \cdot d_{th}$))

Algorithm 2 SDTW

Inputs: $R = r_{1..|R|}$ (reference path); $Q = q_{1..|Q|}$ (query path); d_{th} (success distance threshold).
Outputs: The SDTW score.
  procedure SDTW(R, Q, d_{th})
    if $\delta(R[i], Q[j]) > d_{th}$ **then**
      return 0.0

    return nDTW(R, Q, d_{th})

To compute nDTW, we can define a matrix $\mathbf{C} \in \mathbb{R}^{(|R|+1) \times (|Q|+1)}$ where

$$\mathbf{C}_{i,j} := \text{DTW}(r_{1..i}, q_{1..j}) \tag{3}$$

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 $1 \leq j \leq |Q|$. nDTW can be computed by applying the normalizing operations to $\mathbf{C}[|R|][|Q|]$ and SDTW, as shown in Algorithm 2, limits nDTW to only successful episodes.

DTW, and consequently nDTW and SDTW can be approximated in linear time and space complexity. We refer readers to Salvador and Chan [26] for explanations, proof and pseudo-code. In scenarios where long paths are common, this computational efficiency affords the opportunity to apply nDTW as a evaluation function as well as a reward signal for reinforcement learning agents.
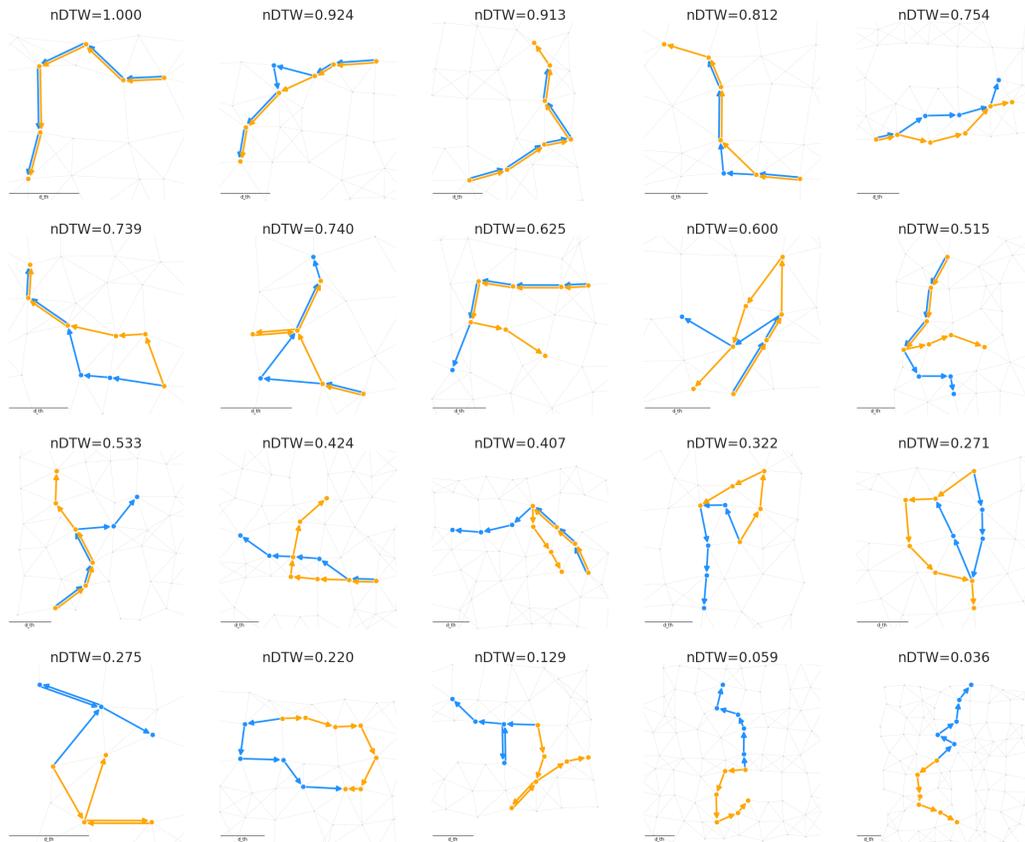
Figure 3: Examples of random reference (blue) and query (orange) paths, sorted by nDTW values.

C   Visualizing nDTW scores

Figure 3 illustrates multiple pairs of reference (blue) and query (orange) paths, accompanied (and sorted) by their nDTW values.