Supplementary Material

A Overview

In this document we describe our algorithm for candidate route generation, and provide analysis on parameters for speaker-driven route selection (pragmatic inference) and other details. We also provide additional qualitative examples showing how pragmatic inference helps instruction following. Finally, we describe our submission to the Vision and Language Navigation Challenge.

B State-Factored Search for Candidate Route Generation

Algorithm B.1 gives pseudocode for the state-factored search algorithm that we use to generate candidate routes for pragmatic inference (outlined in Sec. 3.2 of the main paper).

Algorithm B.1 State-factored search

```
class STATE
    location                    ▷ the agent's physical position in the environment
    completed                   ▷ whether the route has been completed (the STOP action has been taken)
end class

class ROUTE
    states                      ▷ list of STATEs in the route
    score                       ▷ route probability under the follower model, P_F
    expanded                    ▷ whether this route has been expanded in search
end class

function STATEFACTOREDSEARCH(start_state : STATE, K : int)
    ▷ a mapping from STATEs to the best ROUTE ending in that state found so far (whether expanded or unexpanded)
    partial = {}
    ▷ a similar mapping, but containing routes that have been completed
    completed = {}
    start_route = ROUTE([start_state], 1.0, False)
    partial[start_state] = start_route
    candidates = [start_route]
    while |completed| < K and |candidates| > 0 do
        ▷ choose the highest-scoring unexpanded route to expand (route may be complete)
        route = argmax_{r∈candidates} r.score
        route.expanded = True
        ▷ SUCCESSOR generates ROUTEs by taking all possible actions, each of which extends this route by one state, with the resulting total model score, and expanded set to False
        for route' in SUCCESSORS(route) do
            state' = route'.states.last
            cache = completed if route'.completed else partial
            if state' not in cache.keys or cache[state'].score < route'.score then
                cache[state'] = route'
            end if
        end for
        candidates = [route in partial.values if not route.expanded]
    end while
    return completed
end function
```

C Analysis of Inference Parameters and Implementation Details

We further study how varying the speaker weight λ in pragmatic inference (Sec. 3.2 of the main paper) influences the final navigation performance. In Table C.1, we compare our full model (with speaker

#		Validation-Seen			Validation-Unseen		
		NE ↓	SR ↑	OSR ↑	NE ↓	SR ↑	OSR ↑
1	our full model ($\lambda = 0.95$)	**3.08**	70.1	**78.3**	**4.83**	**54.6**	65.2
2	w/o speaker scoring ($\lambda = 0$)	3.17	68.4	74.8	5.94	43.7	53.1
3	w/o state-factoring in search (Sec. B)	3.14	**70.6**	77.4	5.27	50.7	60.7
4	w/o GloVe embedding [1]	3.08	69.6	77.4	4.84	53.2	**66.7**

Table C.1: Effects of speaker scoring and implementation details in our model. NE is navigation error (in meters); lower is better. SR and OSR are success rate and oracle success rate (%); higher is better. Comparison between the 1st and the 2nd row shows that incorporating speaker scoring is crucial to the performance, matching our intuition in Sec. 3.2 of the main paper of the importance of pragmatic inference. Comparison between the 1st and the 3rd row indicates that state-factored search (Sec. B) produces better results than (standard) beam search on val-unseen. Difference between the 1st and the 4th row shows that using GloVe embeddings [1] gives slightly higher success rate.

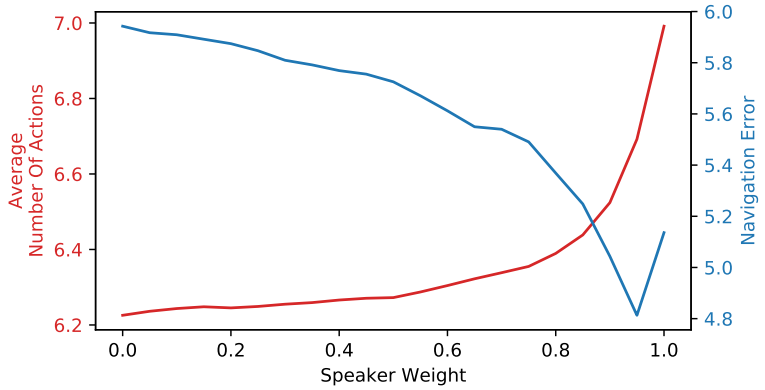


Figure C.1: The average number of actions and navigation error with different speaker weights λ in pragmatic inference (Sec. 3.2 of the main paper), evaluated on the val unseen split. Larger λ results in more number of actions on average, while $\lambda = 0.95$ gives the lowest navigation error.

weight $\lambda = 0.95$) in Row 1 against using only the follower model to score routes ($\lambda = 0$) in Row 2, a baseline that still includes search but does not include speaker scores. The large gap in success rate between $\lambda = 0.95$ and $\lambda = 0$ shows that including speaker scores is crucial to the performance, confirming the importance of pragmatic inference. Figure C.1 shows the average number of actions and the navigation error on val unseen with different speaker weights λ, where $\lambda = 0.95$ gives the lowest navigation error.

In addition, we study how the number of candidate routes, K, used in pragmatic inference impacts the final navigation success rate. Figure C.2 shows the success rate of our model on R2R val seen and val unseen splits using different numbers of candidate routes K for state-factored search (Sec. B). The results show that having more routes to choose between leads to better generalization to the unseen new environments (val unseen), but the gain from increasing the number of candidates tends to saturate quickly. In our final model in Table 2 in the main paper and Table C.1, we use $K = 40$. However, we emphasize that **even with only five route candidates, our model still achieves 50.3% success rate on val unseen**, which improves substantially on both the 35.5% success rate from greedy decoding (i.e. the gold star at $K = 1$ in Figure C.2), as well as the 43.5% success rate given by state-factored search with no pragmatic inference (i.e. the gold triangle at $K = 1$ in Figure C.2).

We also analyze some implementation details in our model. Comparing Row 1 v.s. Row 3 in Table C.1 shows that using state-factored search to produce candidates for pragmatic inference (Sec. B) produces better results on val unseen than using (standard) beam search. Comparing Row 1 v.s. Row 4 in Table C.1 indicates that using GloVe [1] to initialize word embedding sightly improves success rate.

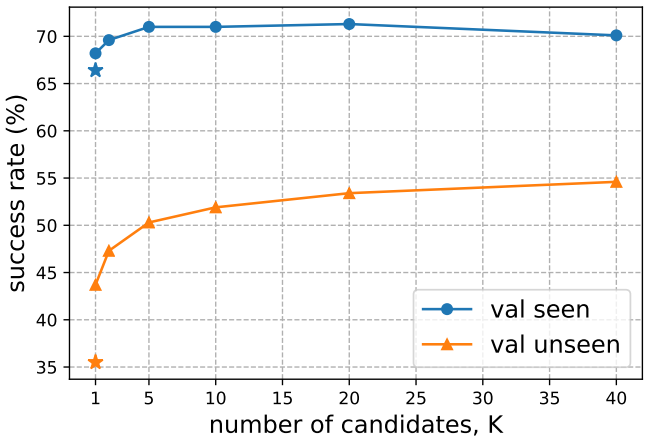


Figure C.2: The success rate of our model on the val seen and val unseen splits, using different numbers K of route candidates (generated by state-factored search) for pragmatic inference. Stars show the performance of greedy inference (without search, and hence without pragmatics). While performance increases with number of candidates up through 40 on val unseen, the success rate tends to saturate. We note improvements both from the state-factored search procedure (comparing the stars to the circle and triangle points at $K = 1$) as well as from having more candidates to choose from in pragmatic inference (comparing larger values of K to smaller).

D Qualitative Examples

We show more examples of how the speaker model helps instruction following on both seen and unseen validation environments. Figure D.3 v.s. D.4 show the step-wise navigation trajectory of the base follower (without pragmatic inference) and the follower model with pragmatic inference, on the val seen split. Figure D.5 v.s. Figure D.6 and Figure D.7 v.s. Figure D.8 show the trajectory of the agent without and with pragmatic inference (using the speaker model) on the val unseen split. The speaker helps disambiguate vague instructions by globally measuring how likely a route can be described by the instruction.

We also visualize the image attention (attention weights $\alpha_{t,i}$ of each view angle i in our panoramic action space in Sec. 3.3 in the main paper), and the textual attention on the input instructions from the sequence-to-sequence model in Figures D.9, D.10 and D.11.

E Submission to Vision and Language Navigation Challenge

We participated in the Vision and Language Navigation Challenge[1], an online challenge for the vision-and-language navigation task on the R2R dataset. We submitted the predictions from our full method to the evaluation server, using single models for the speaker and listener, without any additional ensembling. At the time of writing, our method (under the team name "Speaker-Follower") remains the top-performing method on the challenge leader-board with a success rate of 53.49% (the same success rate as in the Table 2 test split in the main paper).

When generating the predictions for the challenge submission, we modified the method for generating final routes to comply with the challenge guidelines. In Table 1, Table 2 in the main paper and Table C.1, the performance of our full model with pragmatic inference is evaluated using a single top-ranking candidate route, where the candidate routes are generated with state-factored search in Sec. B. Hence, our reported navigation error, success rate and oracle success rate are all computed by choosing a single candidate route per instruction. However, the challenge guidelines require that the submitted trajectories must be generated from *a single independent evaluation run*, where the agent must move sequentially and all its movements during inference must be recorded. So just returning the route selected by pragmatic inference, or by search, would violate the contest guidelines,

[1] `https://evalai.cloudcv.org/web/challenges/challenge-page/97/overview`

as this route may not contain all world states explored during search. On the other hand, the agent is allowed to backtrack to a previous location on its path, as long as the agent does not teleport and all its movements are recorded in the final trajectory (which we confirmed with the challenge organizer).

To comply with the guidelines and maintain physical plausibility, we log all states visited by the search algorithm in the order they are traversed. The agent expands each route one step forward at a time, and then switches to expand the next route. When switching from one route to another route, the agent first backtracks to the closest common ancestor state of the two routes in the search (which could be the starting location). From there it goes to the frontier of the other route and takes an action there to expand it. Once the set of candidate routes has been completed, they are ranked according to Equation 1 in the main paper for pragmatic inference, selecting a route that ends at a target location. Finally, the agent moves from its last visited location in the search to this target location. It then takes the stop action to end the episode. As a result of this, all the agent's movements, including route expansion and route switching, are recorded in its final trajectory.

By design, the sequential inference procedure above yields *exactly the same success rate* as the pragmatic inference procedure, since it returns routes with the same end states. Unsurprisingly, the oracle success rate increases substantially (from 63.9% to 96.0%), since the resulting final trajectory records the visited locations from all routes and the oracle success rate is determined by distance between the ground-truth target location and the closest visited location. We also note that the agent's final trajectory is very long ($1257.38\mathrm{m}$ per instruction on average) since it needs to walk a substantial distance to sequentially expand all candidate routes and to switch between them.

In addition, we also evaluated the predictions from our model without pragmatic inference on the challenge server. Without pragmatic inference, our model achieves 35.08% success rate, 44.45% oracle success rate and $6.62\mathrm{m}$ navigation error on the challenge test set, with an average trajectory length of $14.82\mathrm{m}$. This is close to the performance of our model under the same setting on val unseen split (Row 6 in Table 1 in the main paper).

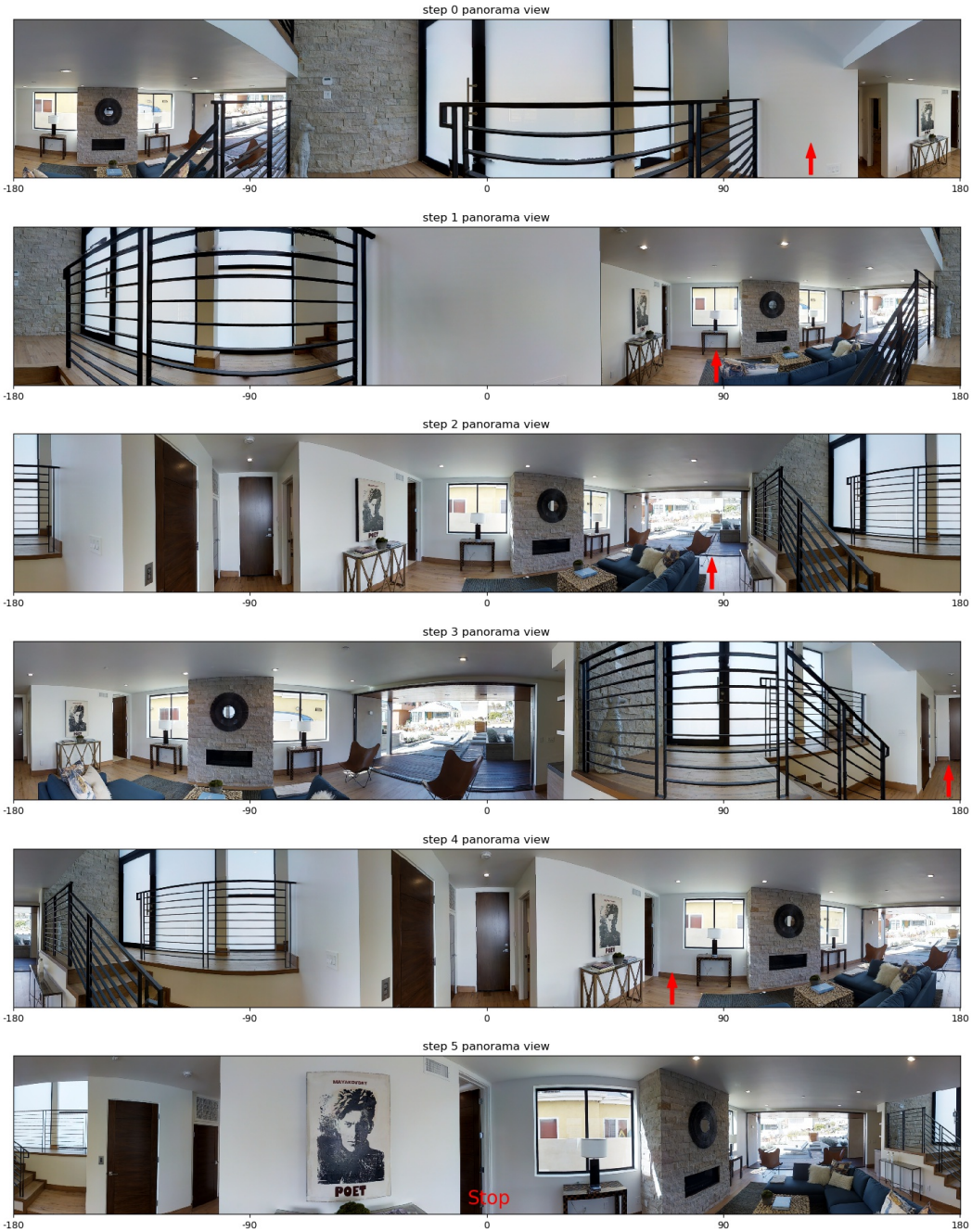
Finally, we note that our method in this work is designed mostly to optimize success rate and navigation error, and we leave potential improvement to reduce trajectory length via inference (such as ordering the routes by location to reduce switching overhead) and modeling (such as building speaker models that can rank partial incomplete routes) to future work.

References

[1] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Instruction:
Walk down and turn right. Walk a bit, and turn right towards the door. Enter inside, and stop in front of a zebra striped rug.

rear: -180 degree *left*: -90 degree *front*: 0 degree *right*: +90 degree *rear*: +180 degree

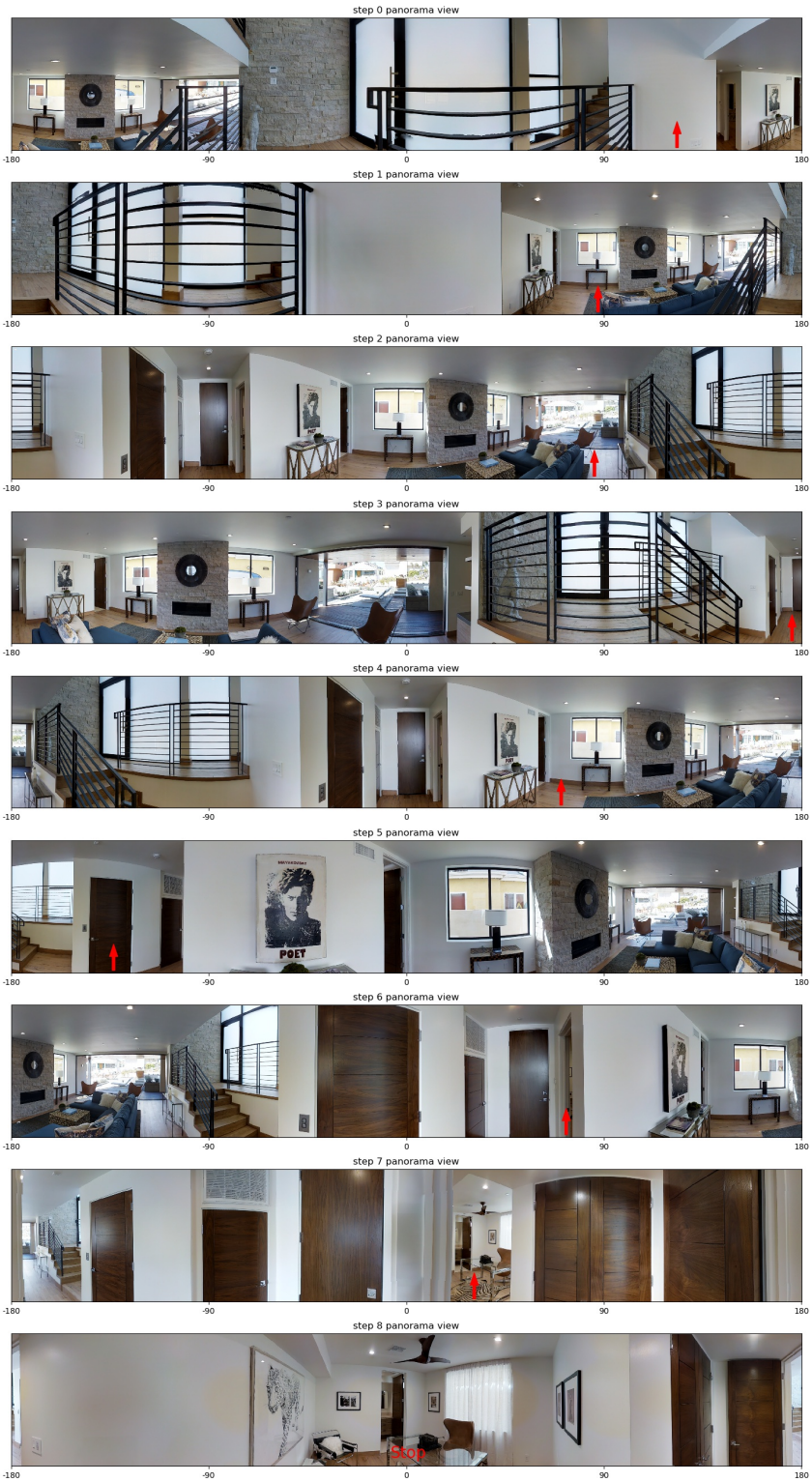


Navigation steps of the panorama agent. The red arrow shows the direction chosen by the agent to go next.

Figure D.3: Follower **without pragmatic inference** on val seen. The instruction involves an ambiguous "walk a bit" command. Without pragmatic reasoning by the speaker, the follower failed to predict how much to move forward, stopping at a wrong location without entering the door.

Instruction:
Walk down and turn right. Walk a bit, and turn right towards the door. Enter inside, and stop in front of a zebra striped rug.

rear: -180 degree *left*: -90 degree *front*: 0 degree *right*: +90 degree *rear*: +180 degree

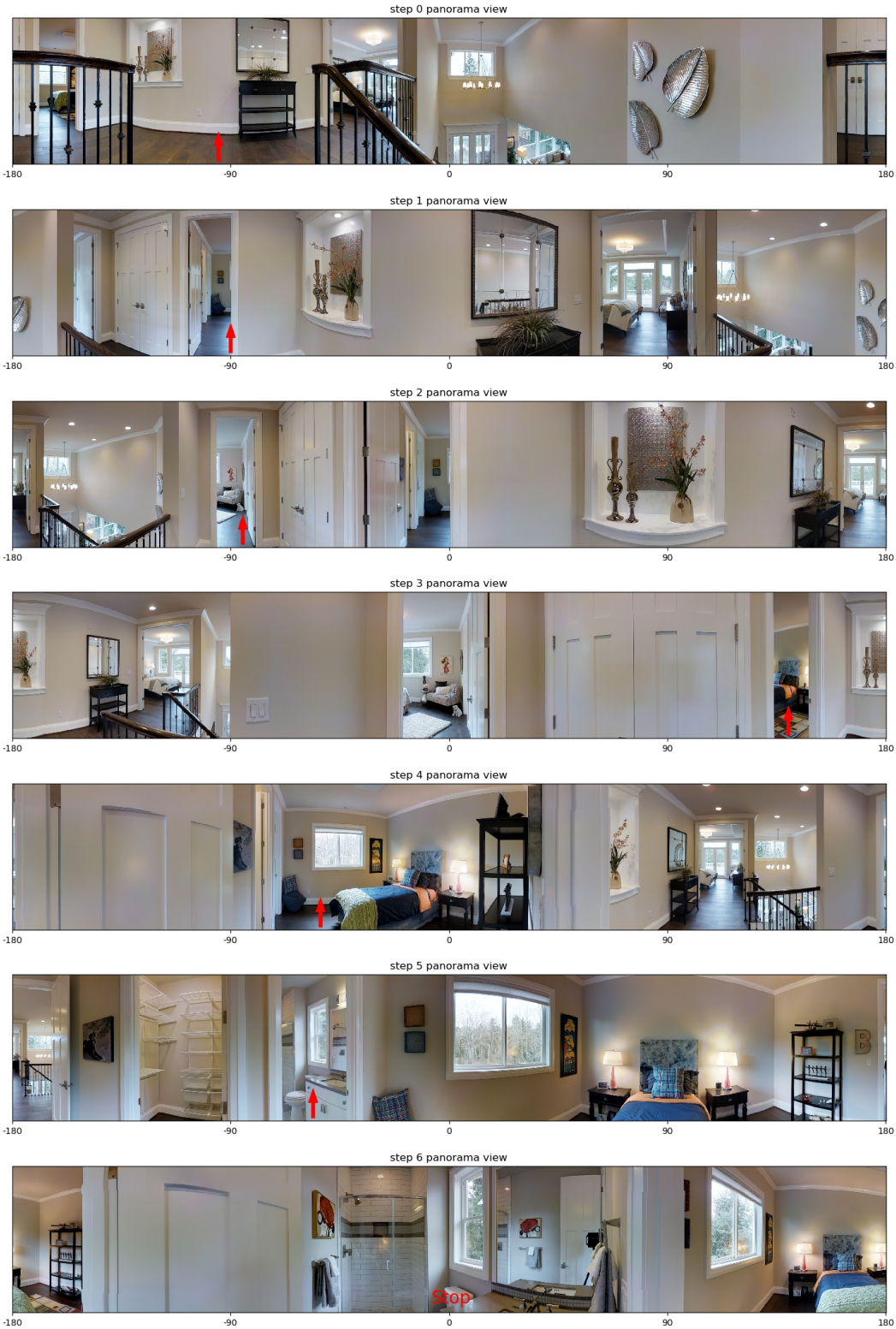


Navigation steps of the panorama agent. The red arrow shows the direction chosen by the agent to go next.

Figure D.4: Follower **with pragmatic inference** on val seen. With the help of the speaker, the follower could disambiguate "walk a bit" to move the right amount to the correct location. It then turned right and walked into the door to stop by the "zebra stripped rug".

Instruction:
Walk past hall table. Walk into bedroom. Make left at table clock. Wait at bathroom door threshold.

rear: -180 degree *left*: -90 degree *front*: 0 degree *right*: +90 degree *rear*: +180 degree



Navigation steps of the panorama agent. The red arrow shows the direction chosen by the agent to go next.

Figure D.5: Follower **without pragmatic inference** on val unseen. The command "walk into bedroom" is ambiguous since there are two bedrooms (one on the left and one on the right). The follower could not decide which bedroom to enter, but went into the wrong room with no "table clock".

Instruction:
Walk past hall table. Walk into bedroom. Make left at table clock. Wait at bathroom door threshold.

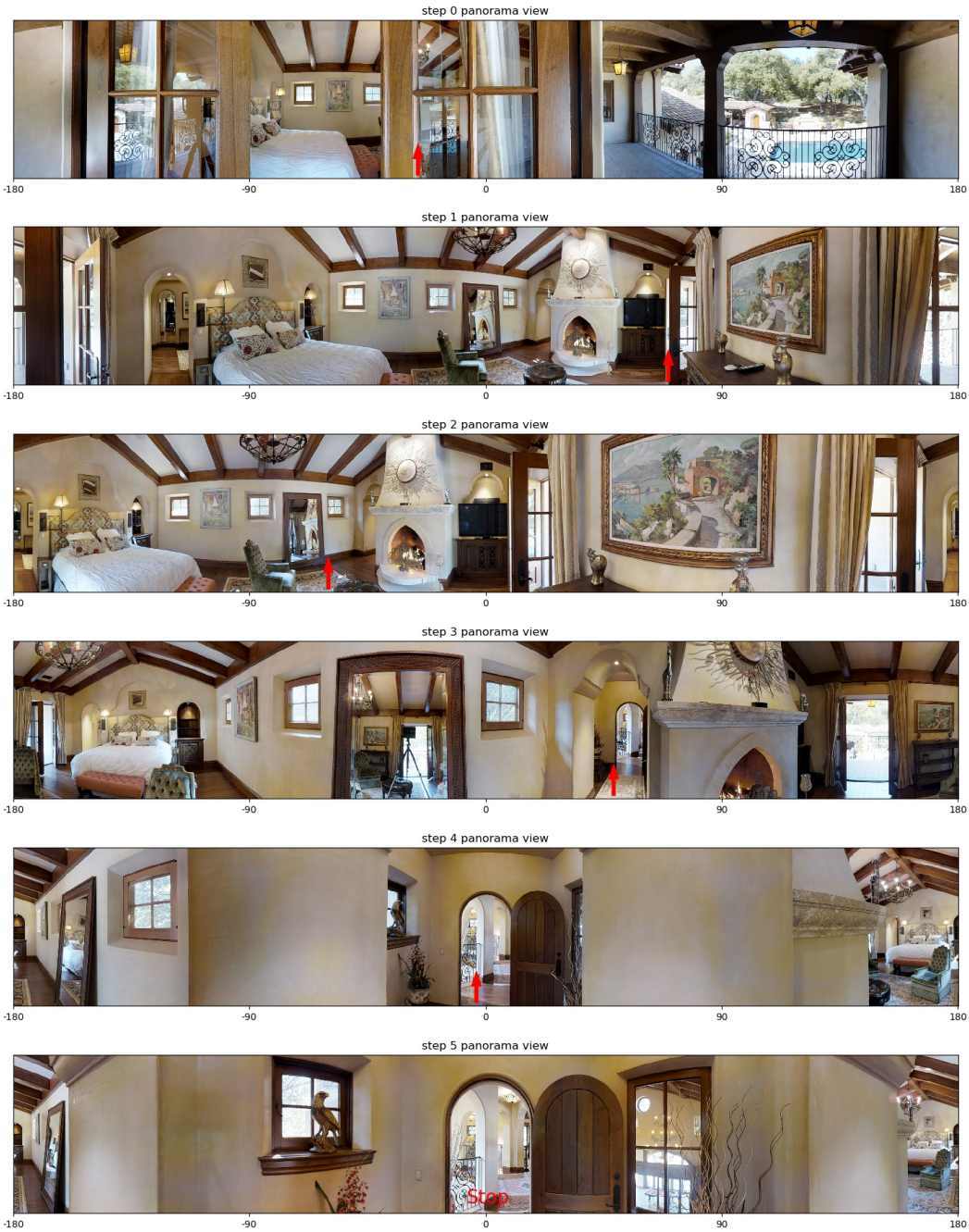
rear: -180 degree *left*: -90 degree *front*: 0 degree *right*: +90 degree *rear*: +180 degree

Navigation steps of the panorama agent. The red arrow shows the direction chosen by the agent to go next.

Figure D.6: Follower **with pragmatic inference** on val unseen. The speaker model helps resolve the ambiguous "walk into bedroom" command (there are two bedrooms), allowing the follower to enter the correct bedroom on the right, where it could see a "table clock".

Instruction:
Enter the bedroom and make a slight right. Walk across the room near the foot of the bed. Turn right at the end of the rug. Wait near the mirror.

rear: -180 degree *left*: -90 degree *front*: 0 degree *right*: +90 degree *rear*: +180 degree



Navigation steps of the panorama agent. The red arrow shows the direction chosen by the agent to go next.

Figure D.7: Follower **without pragmatic inference** on val unseen. Although making a right turn as described, the follower fails to turn right at the correct location, and stopped at the door instead of the mirror. The route taken by the follower would be better described as *"...wait near the door"* by a human, which the speaker could learn to capture.

Instruction:
Enter the bedroom and make a slight right. Walk across the room near the foot of the bed. Turn right at the end of the rug. Wait near the mirror.

rear: -180 degree *left*: -90 degree *front*: 0 degree *right*: +90 degree *rear*: +180 degree

Navigation steps of the panorama agent. The red arrow shows the direction chosen by the agent to go next.

Figure D.8: Follower **with pragmatic inference** on val unseen. Using the speaker to measure how likely a route matches the provided description, the follower made the right turn at the correct location *"the end of the rug"*, and stopped near the mirror.

Instruction: *Go through the dooorway on the right, continue straightacross the hallway and into the room ahead. Stop near the shell.*

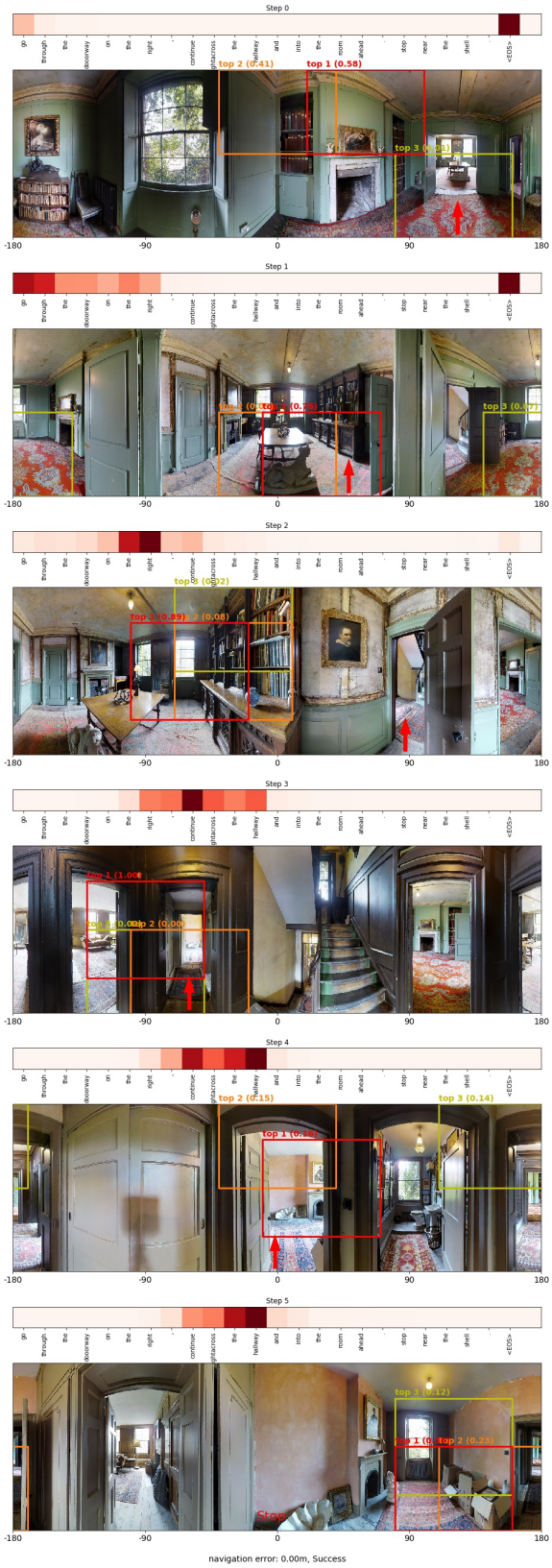


Figure D.9: Image and textual attention visualization on val unseen (best viewed at 200%). At each step, the textual attention is shown at the top, and the 1st, 2nd and 3rd most attended view angles are shown in red, orange and yellow boxes, respectively (the number in the parenthesis shows the attention weight). The red arrow shows the direction chosen by the agent to go next.

Instruction: *Walk through the kitchen, enter the dining room, walk to the doorway to the right of the dining room table, wait at the glass table.*

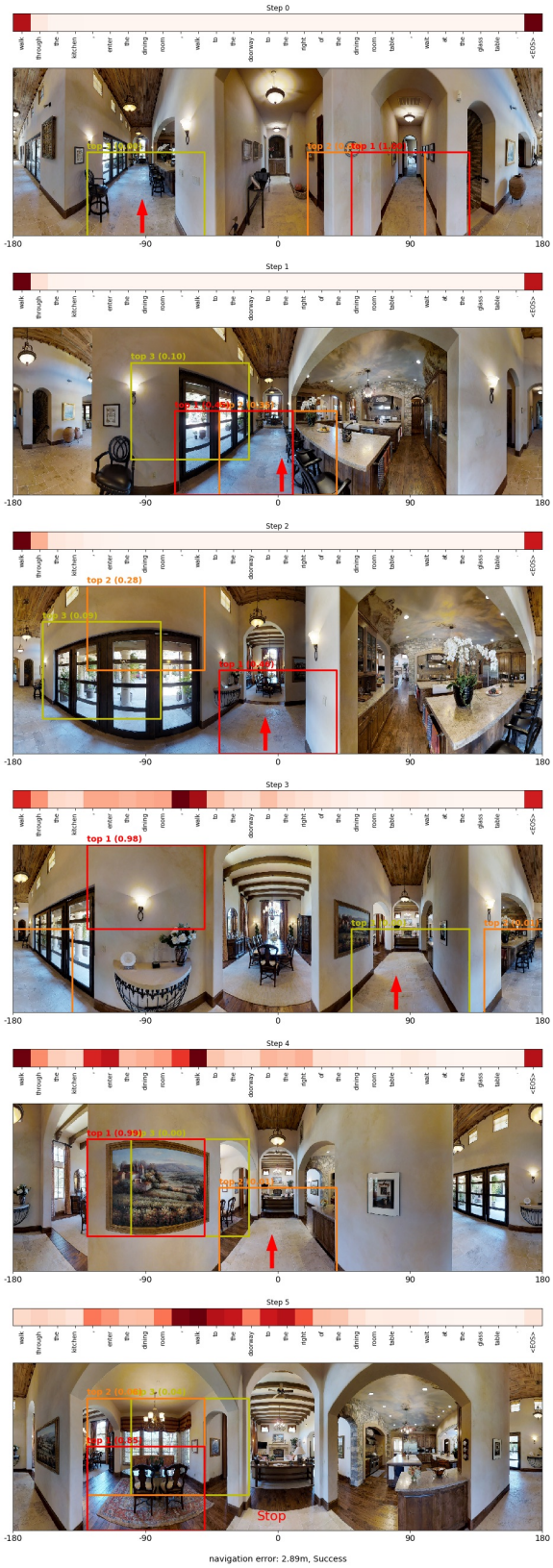


Figure D.10: Image and textual attention visualization on val unseen (best viewed at 200%). At each step, the textual attention is shown at the top, and the 1st, 2nd and 3rd most attended view angles are shown in red, orange and yellow boxes, respectively (the number in the parenthesis shows the attention weight). The red arrow shows the direction chosen by the agent to go next.

Instruction: *Walk up stairs. Turn left and walk to the double doors by the living room.*

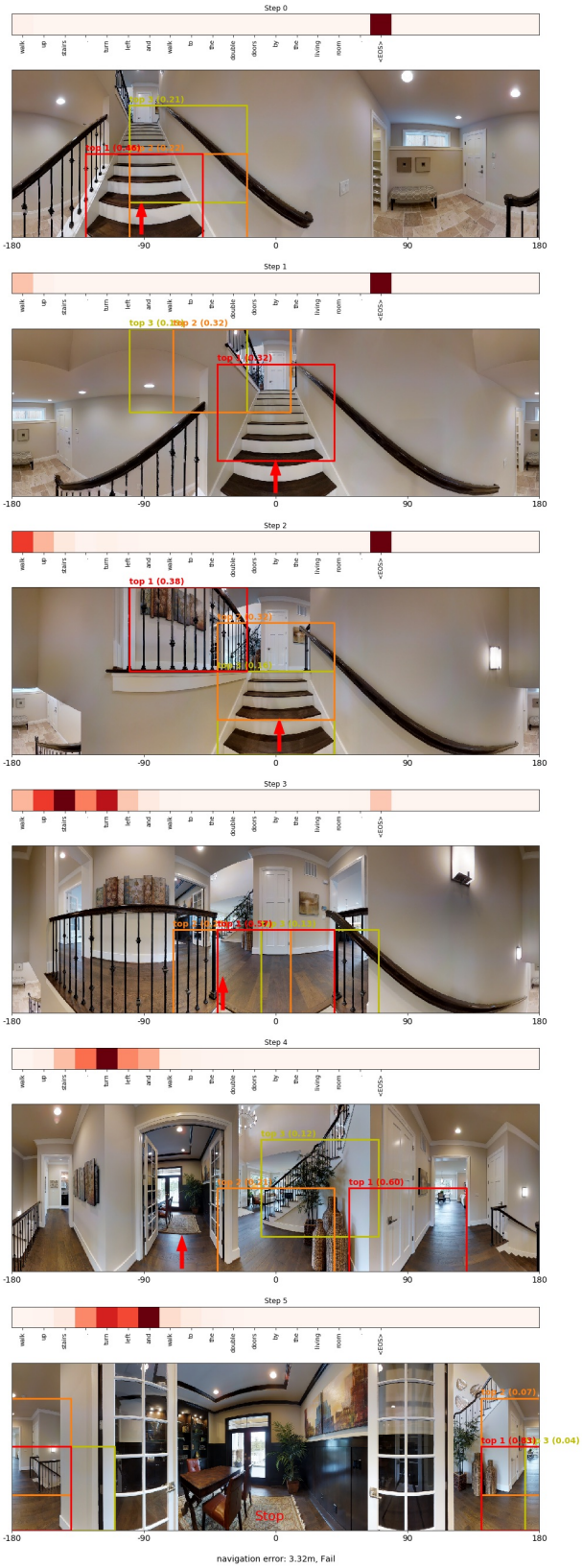


Figure D.11: Image and textual attention visualization on val unseen (best viewed at 200%). At each step, the textual attention is shown at the top, and the 1st, 2nd and 3rd most attended view angles are shown in red, orange and yellow boxes, respectively (the number in the parenthesis shows the attention weight). The red arrow shows the direction chosen by the agent to go next.