TOUCHDOWN: Natural Language Navigation and Spatial Reasoning in Visual Street Environments

Howard Chen*
ASAPP Inc.
New York, NY
hchen@asapp.com

Alane Suhr Dipendra Misra Noah Snavely Yoav Artzi
Department of Computer Science & Cornell Tech, Cornell University
New York, NY
{suhr, dkm, snavely, yoav}@cs.cornell.edu

Abstract

We study the problem of jointly reasoning about language and vision through a navigation and spatial reasoning task. We introduce the TOUCHDOWN *task and dataset, where an agent must first follow navigation instructions in a real-life visual urban environment, and then identify a location described in natural language to find a hidden object at the goal position. The data contains* 9,326 *examples of English instructions and spatial descriptions paired with demonstrations. Empirical analysis shows the data presents an open challenge to existing methods, and qualitative linguistic analysis shows that the data displays richer use of spatial reasoning compared to related resources. The environment and data are available at https://touchdown.ai.*

1. Introduction

Consider the visual challenges of following natural language instructions in a busy urban environment. Figure 1 illustrates this problem. The agent must identify objects and their properties to resolve mentions to *traffic light* and *American flags*, identify patterns in how objects are arranged to find the *flow of traffic*, and reason about how the relative position of objects changes as it moves to *go past* objects. Reasoning about vision and language has been studied extensively with various tasks, including visual question answering [3, 34], visual navigation [2, 25], interactive question answering [9, 12], and referring expression resolution [16, 22, 23]. However, existing work has largely focused on relatively simple visual input, including object-focused photographs [20, 28] or simulated environments [4, 9, 19, 25, 33]. While this has enabled significant progress in visual understanding, the use of real-world visual input not only increases the challenge of the vision task, it also drastically changes the kind of language it elicits and requires fundamentally different reasoning.

Turn and go with the flow of traffic. At the first traffic light turn left. Go past the next two traffic light, As you come to the third traffic light you will see a white building on your left with many American flags on it. Touchdown is sitting in the stars of the first flag.

Figure 1. An illustration of the task. The agent follows the instructions to reach the goal, starting by re-orientating itself (top image) and continuing by moving through the streets (two middle images). At the goal (bottom), the agent uses the spatial description (underlined) to locate Touchdown the bear. Touchdown only appears if the guess is correct (see bottom right detail).

In this paper, we study the problem of reasoning about vision and natural language using an interactive visual navigation environment based on Google Street View.[1] We design the task of first following instructions to reach a goal

*Work done at Cornell University.

[1] https://developers.google.com/maps/documentation/streetview/intro

position, and then resolving a spatial description at the goal by identifying the location in the observed image of Touchdown, a hidden teddy bear. Using this environment and task, we release TOUCHDOWN,[2] a dataset for navigation and spatial reasoning with real-life observations.

We design our task for diverse use of spatial reasoning, including for following instructions and resolving the spatial descriptions. Navigation requires the agent to reason about its relative position to objects and how these relations change as it moves through the environment. In contrast, understanding the description of the location of Touchdown requires the agent to reason about the spatial relations between observed objects. The two tasks also diverge in their learning challenges. While in both learning requires relying on indirect supervision to acquire spatial knowledge and language grounding, for navigation, the training data includes demonstrated actions, and for spatial description resolution, annotated target locations. The task can be addressed as a whole, or decomposed to its two portions.

The key data collection challenge is designing a scalable process to obtain natural language data that reflects the richness of the visual input while discouraging overly verbose and unnatural language. In our data collection process, workers write and follow instructions. The writers navigate in the environment and hide Touchdown. Their goal is to make sure the follower can execute the instruction to find Touchdown. The measurable goal allows us to reward effective writers, and discourages overly verbose descriptions.

We collect 9,326 examples of the complete task, which decompose to the same number of navigation tasks and 27,575 spatial description resolution (SDR) tasks. Each example is annotated with a navigation demonstration and the location of Touchdown. Our linguistically-driven analysis shows the data requires significantly more complex reasoning than related datasets. Nearly all examples require resolving spatial relations between observable objects and between the agent and its surroundings, and each example contains on average 5.3 commands and refers to 10.7 unique entities in its environment.

We empirically study the navigation and SDR tasks independently. For navigation, we focus on the performance of existing models trained with supervised learning. For SDR, we cast the problem of identifying Touchdown's location as an image feature reconstruction problem using a language-conditioned variant of the UNET architecture [29, 25]. This approach significantly outperforms several strong baselines.

2. Related Work and Datasets

Jointly reasoning about vision and language has been studied extensively, most commonly focusing on static visual input for reasoning about image captions [20, 8, 28, 31, 32] and grounded question answering [3, 13, 34]. Recently, the problem has been studied in interactive simulated environments where the visual input changes as the agent acts, such as interactive question answering [9, 12,] and instruction following [25, 26]. In contrast, we focus on an interactive environment with real-world observations.

The most related resources to ours are R2R [2] and Talk the Walk [10]. R2R uses panorama graphs of house environments for the task of navigation instruction following. It includes 90 unique environments, each containing an average of 119 panoramas, significantly smaller than our 29,641 panoramas. Our larger environment requires following the instructions closely, as finding the goal using search strategies is unlikely, even given a large number of steps. We also observe that the language in our data is significantly more complex than in R2R (Section 5). Our environment setup is related to Talk the Walk, which uses panoramas in small urban environments for a navigation dialogue task. In contrast to our setup, the instructor does not observe the panoramas, but instead sees a simplified diagram of the environment with a small set of pre-selected landmarks. As a result, the instructor has less spatial information compared to TOUCHDOWN. Instead the focus is on conversational coordination.

SDR is related to the task of referring expression resolution, for example as studied in ReferItGame [16] and Google Refexp [22]. Referring expressions describe an observed object, mostly requiring disambiguation between the described object and other objects of the same type. In contrast, the goal of SDR is to describe a specific location rather than discriminating. This leads to more complex language, as illustrated by the comparatively longer sentences of SDR (Section 5). Kitaev and Klein [18] proposed a similar task to SDR, where given a spatial description and a small set of locations in a fully-observed simulated 3D environment, the system must select the location described from the set. We do not use distractor locations, requiring a system to consider all areas of the image to resolve a spatial description.

3. Environment and Tasks

We use Google Street View to create a large navigation environment. Each position includes a 360° RGB panorama. The panoramas are connected in a graph-like structure with undirected edges connecting neighboring panoramas. Each edge connects to a panorama in a specific heading. For each panorama, we render perspective images for all headings that have edges. Our environment includes 29,641 panoramas and 61,319 edges from New York City. Figure 2 illustrates the environment.

We design two tasks: navigation and spatial description resolution (SDR). Both tasks require recognizing objects and the spatial relations between them. Navigation focuses on egocentric spatial reasoning, where instructions refer to the agent's relationship with its environment, including the

[2] Touchdown is the unofficial mascot of Cornell University.

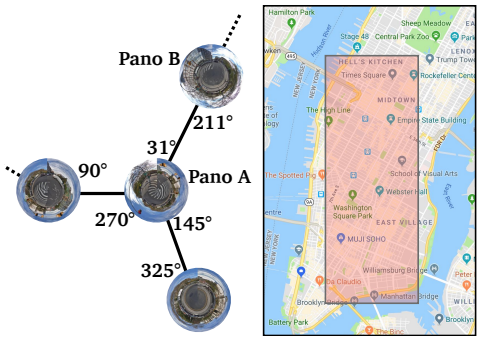


Figure 2. An illustration of the environment. Left: part of the graph structure with polarly projected panoramas illustrating positions linked by edges, each labeled with its heading. Heading angles shown closer to each panorama represent the outgoing angle from that panorama; for example, the heading from Pano A to Pano B is 31°. Right: the area in New York City covered by the graph.

objects it observes. The SDR task displays more allocentric reasoning, where the language requires understanding the relations between the observed objects to identify the target location. While navigation requires generating a sequence of actions from a small set of possible actions, SDR requires choosing a specific pixel in the observed image. Both tasks present different learning challenges. The navigation task could benefit from reward-based learning, while the SDR task defines a supervised learning problem. The two tasks can be addressed separately, or combined by completing the SDR task at the goal position at the end of the navigation.

3.1. Navigation

The agent's goal is to follow a natural language instruction and reach a goal position. Let $\mathcal{S}$ be the set of all states. A state $s \in \mathcal{S}$ is a pair $(\mathbf{I}, \alpha)$, where $\mathbf{I}$ is a panorama and α is the heading angle indicating the agent heading. We only allow states where there is an edge connecting to a neighboring panorama in the heading α. Given a navigation instruction $\bar{x}_n$ and a start state $s_1 \in \mathcal{S}$, the agent performs a sequence of actions. The set of actions $\mathcal{A}$ is $\{\texttt{FORWARD}, \texttt{LEFT}, \texttt{RIGHT}, \texttt{STOP}\}$. Given a state s and an action $a \in \mathcal{A}$, the state is deterministically updated using a transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$. The `FORWARD` action moves the agent along the edge in its current heading. Formally, if the environment includes the edge $(\mathbf{I}_i, \mathbf{I}_j)$ at heading α in $\mathbf{I}_i$, the transition is $T((\mathbf{I}_i, \alpha), \texttt{FORWARD}) = (\mathbf{I}_j, \alpha')$. The new heading α' is the heading of the edge in $\mathbf{I}_j$ with the closest heading to α. The `LEFT` (`RIGHT`) action changes the agent heading to the heading of the closest edge on the left (right). Formally, if the position panorama $\mathbf{I}$ has edges at headings $\alpha > \alpha' > \alpha''$, $T((\mathbf{I}, \alpha), \texttt{LEFT}) = (\mathbf{I}, \alpha')$ and $T((\mathbf{I}, \alpha), \texttt{RIGHT}) = (\mathbf{I}, \alpha'')$. Given a start state s_1 and a navigation instruction $\bar{x}_n$, an execution $\bar{e}$ is a sequence of state-action pairs $\langle (s_1, a_1), ..., (s_m, a_m) \rangle$, where $T(s_i, a_i) = s_{i+1}$ and $a_m = \texttt{STOP}$.

Evaluation We use three evaluation metrics: task completion, shortest-path distance, and success-weighted edit distance. Task completion (TC) measures the accuracy of completing the task correctly. We consider an execution correct if the agent reaches the exact goal position or one of its neighboring nodes in the environment graph. Shortest-path distance (SPD) measures the mean distance in the graph between the agent's final panorama and the goal. SPD ignores turning actions and the agent heading. Success weighted by edit distance (SED) is $\frac{1}{N} \sum_{i=1}^{N} S_i (1 - \frac{\text{lev}(\bar{e}, \hat{\bar{e}})}{\max(|\bar{e}|, |\hat{\bar{e}}|)})$, where the summation is over N examples, S_i is a binary task completion indicator, $\bar{e}$ is the reference execution, $\hat{\bar{e}}$ is the predicted execution, $\text{lev}(\cdot, \cdot)$ is the Levenshtein edit distance, and $|\cdot|$ is the execution length. The edit distance is normalized and inversed. We measure the distance and length over the sequence of panoramas in the execution, and ignore changes of orientation. SED is related to success weighted by path length (SPL) [1], but is designed for instruction following in graph-based environments, where a specific correct path exists.

3.2. Spatial Description Resolution (SDR)

Given an image $\mathbf{I}$ and a natural language description $\bar{x}_s$, the task is to identify the point in the image that is referred to by the description. We instantiate this task as finding the location of Touchdown, a teddy bear, in the environment. Touchdown is hidden and not visible in the input. The image $\mathbf{I}$ is a 360° RGB panorama, and the output is a pair of (x, y) coordinates specifying a location in the image.

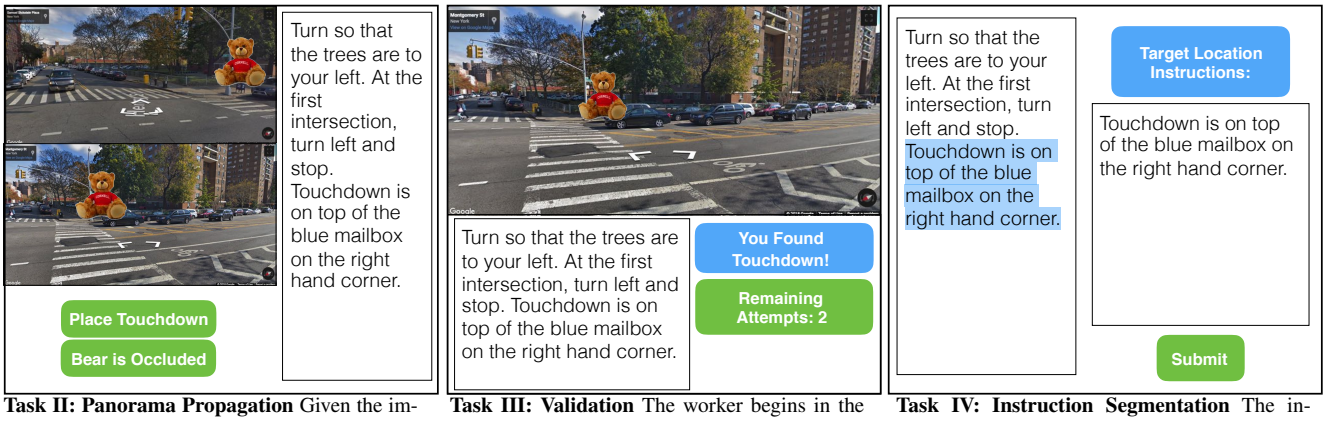
Evaluation We use three evaluation metrics: accuracy, consistency, and distance error. Accuracy is computed with regard to an annotated location. We consider a prediction as correct if the coordinates are within a slack radius of the annotation. We measure accuracy for radiuses of 40, 80, and 120 pixels and use euclidean distance. Our data collection process results in multiple images for each sentence. We use this to measure consistency over unique sentences, which is measured similar to accuracy, but with a unique sentence considered correct only if all its examples are correct [11]. We compute consistency for each slack value. We also measure the mean euclidean distance between the annotated location and the predicted location.

4. Data Collection

We frame the data collection process as a treasure-hunt task where a leader hides a treasure and writes directions to find it, and a follower follows the directions to find the treasure. The process is split into four crowdsourcing tasks (Figure 3). The two main tasks are writing and following. In the writing task, a leader follows a prescribed route and hides Touchdown the bear at the end, while writing instructions that describe the path and how to find Touchdown.

Task I: Instruction Writing The worker starts at the beginning of the route facing north (a). The prescribed route is shown in the overhead map (bottom left of each image). The worker faces the correct direction and follows the path, while writing instructions that describe these actions (b). After following the path, the worker reaches the goal position, places Touchdown, and completes writing the instructions (c).

(a) (b) (c)



Task II: Panorama Propagation Given the image from the leader's final position (top), including Touchdown's placement, and the instructions (right), the worker annotates the location of Touchdown in the neighboring image (bottom).

Task III: Validation The worker begins in the same heading as the leader, and follows the instructions (bottom left) by navigating the environment. When the worker believes they have reached the goal, they guess the target location by clicking in the Street View image.

Task IV: Instruction Segmentation The instructions are shown (left). The worker highlights segments corresponding to the navigation and target location subtasks. The highlighted segment is shown to the worker (right).

Figure 3. Illustration of the data collection process.

The following task requires following the instructions from the same starting position to navigate and find Touchdown. Additional tasks are used to segment the instructions into the navigation and target location tasks, and to propagate Touchdown's location to panoramas that neighbor the final panorama. We use a customized Street View interface for data collection. However, the final data uses a static set of panoramas that do not require the Street View interface.

Task I: Instruction Writing We generate routes by sampling start and end positions. The sampling process results in routes that often end in the middle of a city block. This encourages richer language, for example by requiring to describe the goal position rather than simply directing to the next intersection. The route generation details are described in the Supplementary Material. For each task, the worker is placed at the starting position facing north, and asked to follow a route specified in an overhead map view to a goal position. Throughout, they write instructions describing the path. The initial heading requires the worker to re-orient to the path, and thereby familiarize with their surroundings better. It also elicits interesting re-orientation instructions that often include references to the direction of objects (e.g., *flow of traffic*) or their relation to the agent (e.g., *the umbrellas are to the right*). At the goal panorama, the worker is asked to place Touchdown in a location of their choice that is not a moving object (e.g., a car or pedestrian) and to describe the location in their instructions. The worker goal is to write instructions that a human follower can use to correctly navigate and locate the target without knowing the correct path or location of Touchdown. They are not permitted to write instructions that refer to text in the images, including street names, store names, or numbers.

Task II: Target Propagation to Panoramas The writing task results in the location of Touchdown in a single panorama in the Street View interface. However, resolving the spatial description to the exact location is also possible from neighboring panoramas where the target location is visible. We use a crowdsourcing task to propagate the loca-

Orient yourself in the direction of the red ladder. Go straight and take a left at the intersection with islands. Take another left at the intersection with a gray trash can to the left. Go straight until ***near the end of the fenced in playground and court to the right near the end of the fenced in playground and court to the right. Touchdown is on the last basketball hoop to the right.***

Figure 4. Example instruction where the annotated navigation (underlined) and SDR (bolded) segments overlap.

Task	Number of Workers
Instruction Writing	224
Target Propagation	218
Validation	291
Instruction Segmentation	46

Table 1. Number of workers who participated in each task.

tion of Touchdown to neighboring panoramas in the Street View interface, and to the identical panoramas in our static data. This allows to complete the task correctly even if not stopping at the exact location, but still reaching a semantically equivalent position. The propagation in the Street View interface is used for our validation task. The task includes multiple steps. At each step, we show the instruction text and the original Street View panorama with Touchdown placed, and ask for the location for a single panorama, either from the Street View interface or from our static images. The worker can indicate if the target is occluded. The propagation annotation allows us to create multiple examples for each SDR, where each example uses the same SDR but shows the environment from a different position.

Task III: Validation We use a separate task to validate each instruction. The worker is asked to follow the instruction in the customized Street View interface and find Touchdown. The worker sees only the Street View interface, and has no access to the overhead map. The task requires navigation and identifying the location of Touchdown. It is completed correctly if the follower clicks within a 90-pixel radius[3] of the ground truth target location of Touchdown. This requires the follower to be in the exact goal panorama, or in one of the neighboring panoramas we propagated the location to. The worker has five attempts to find Touchdown. Each attempt is a click. If the worker fails, we create another task for the same example to attempt again. If the second worker fails as well, the example is discarded.

Task IV: Segmentation We annotate each token in the instruction to indicate if it describes the navigation or SDR tasks. This allows us to address the tasks separately. First, a worker highlights a consecutive prefix of tokens to indicate the navigation segment. They then highlight a suffix of tokens for the SDR task. The navigation and target location segments may overlap (Figure 4).

Workers and Qualification We require passing a qualification task to do the writing task. The qualifier task requires correctly navigating and finding Touchdown for a predefined set of instructions. We consider workers that succeed in three out of the four tasks as qualified. The other three tasks do not require qualification. Table 1 shows how many workers participated in each task.

Payment and Incentive Structure The base pay for instruction writing is $0.60. For target propagation, validation, and segmentation we paid $0.15, $0.25, and $0.12. We incentivize the instruction writers and followers with a bonus system. For each instruction that passes validation, we give the writer a bonus of $0.25 and the follower a bonus of $0.10. Both sides have an interest in completing the task correctly. The size of the graph makes it difficult, and even impossible, for the follower to complete the task and get the bonus if the instructions are wrong.

[3]This is roughly the size of Touchdown. The number is not directly comparable to the SDR accuracy measures due to different scaling.

Dataset	Dataset Size	Vocab. Size	Mean Text Length	Real Vision?
TOUCHDOWN	9,326	5,625	108.0	
Navigation	9,326	4,999	89.6	✓
SDR	25,575	3,419	29.7	
R2R [2]	21,567	3,156	29.3	✓
SAIL [21]	706	563	36.7	✗
LANI [25]	5,487	2,292	61.9	✗

Table 2. Data statistics of TOUCHDOWN, compared to related corpora. For TOUCHDOWN, we report statistics for the complete task, navigation only, and SDR only. Vocabulary size and text length are computed on the combined training and development sets. SAIL and LANI statistics are computed using paragraph data.

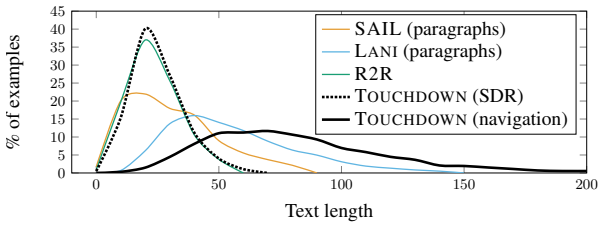


Figure 5. Text lengths in TOUCHDOWN and related corpora.

5. Data Statistics and Analysis

Workers completed 11,019 instruction-writing tasks, and 12,664 validation tasks. 89.1% examples were correctly validated, 80.1% on the first attempt and 9.0% on the second.[4] While we allowed five attempts at finding Touchdown during validation tasks, 64% of the tasks required a single attempt. The value of additional attempts decayed quickly: only 1.4% of the tasks were only successful after five attempts. For the full task and navigation-only, TOUCHDOWN includes 9,326 examples with 6,526 in the training set, 1,391 in the development set, and 1,409 in the test set. For the SDR task, TOUCHDOWN includes 9,326 unique descriptions and 25,575 examples with 17,880 for training, 3,836 for development, and 3,859 for testing. We use our

[4]Several paths were discarded due to updates in Street View data.

initial paths as gold-standard demonstrations, and the placement of Touchdown by the original writer as the reference location. Table 2 shows basic data statistics. The mean instruction length is 108.0 tokens. The average overlap between navigation and SDR is 11.4 tokens. Figure 5 shows the distribution of text lengths. Overall, TOUCHDOWN contains a larger vocabulary and longer navigation instructions than related corpora. The paths in TOUCHDOWN are longer than in R2R [2], on average 35.2 panoramas compared to 6.0. SDR segments have a mean length of 29.8 tokens, longer than in common referring expression datasets; ReferItGame [16] expressions 4.4 tokens on average and Google RefExp [22] expressions are 8.5.

We perform qualitative linguistic analysis of TOUCHDOWN to understand the type of reasoning required to solve the navigation and SDR tasks. We identify a set of phenomena, and randomly sample 25 examples from the development set, annotating each with the number of times each phenomenon occurs in the text. Table 3 shows results comparing TOUCHDOWN with R2R.[5] Sentences in TOUCHDOWN refer to many more unique, observable entities (10.7 vs 3.7), and almost all examples in TOUCHDOWN include coreference to a previously-mentioned entity. More examples in TOUCHDOWN require reasoning about counts, sequences, comparisons, and spatial relationships of objects. Correct execution in TOUCHDOWN requires taking actions only when certain conditions are met, and ensuring that the agent's observations match a described scene, while this is rarely required in R2R. Our data is rich in spatial reasoning. We distinguish two types: between multiple objects (*allocentric*) and between the agent and its environment (*egocentric*). We find that navigation segments contain more egocentric spatial relations than SDR segments, and SDR segments require more allocentric reasoning. This corresponds to the two tasks: navigation mainly requires moving the agent relative to its environment, while SDR requires resolving a point in space relative to other objects.

6. Spatial Reasoning with LINGUNET

We cast the SDR task as a language-conditioned image reconstruction problem, where we predict a distribution of the location of Touchdown over the entire observed image.

6.1. Model

We use the LINGUNET architecture [25, 5], which was originally introduced for goal prediction and planning in instruction following. LINGUNET is a language-conditioned variant of the UNET architecture [29], an image-to-image encoder-decoder architecture widely used for image segmentation. LINGUNET incorporates language into the image reconstruction phase to fuse the two modalities. We modify the architecture to predict a probability distribution over the input panorama image.

[5] See the Supplementary Material for analysis of SAIL and LANI.

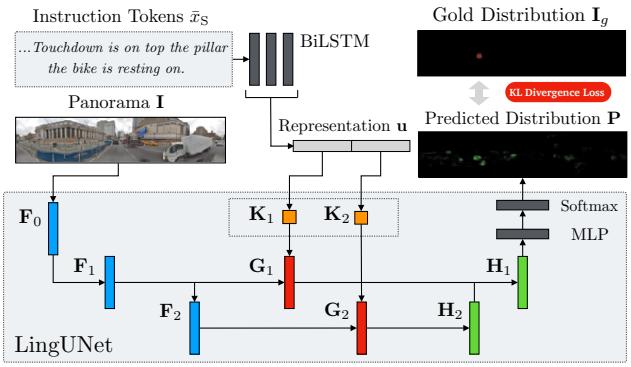


Figure 6. LingUNet architecture with two layers ($m = 2$).

We process the description text tokens $\bar{x}_s = \langle x_1, x_2, \ldots, x_l \rangle$ using a bi-directional Long Short-term Memory (LSTM) recurrent neural network to generate l hidden states. The forward computation is $\mathbf{h}_i^f = \text{BiLSTM}(\varphi(x_i), \mathbf{h}_{i-1}^f), i = 1, \ldots, l$, where φ is a learned word embedding function. We compute the backward hidden states $\mathbf{h}_i^b$ similarly. The text representation is an average of the concatenated hidden states $\mathbf{x} = \frac{1}{l} \sum_{i=1}^{l} [h_i^f; h_i^b]$. We map the RGB panorama $\mathbf{I}$ to a feature representation $\mathbf{F}_0$ with a pre-trained RESNET18 [14].

LINGUNET performs m levels of convolution and deconvolution operations. We generate a sequence of feature maps $\mathbf{F}_k = \text{CNN}_k(\mathbf{F}_{k-1}), k = 1, \ldots, m$ with learned convolutional layers CNN_k. We slice the text representation $\mathbf{x}$ to m equal-sized slices, and reshape each with a linear projection to a 1×1 filter $\mathbf{K}_k$. We convolve each feature map $\mathbf{F}_k$ with $\mathbf{K}_k$ to obtain a text-conditioned feature map $\mathbf{G}_k = \text{CONV}(\mathbf{K}_k, \mathbf{F}_k)$. We use m deconvolution operations to generate feature maps of increasing size to create $\mathbf{H}_1$:

$$\mathbf{H}_k = \begin{cases} \text{DECONV}_k([\mathbf{H}_{k+1}; \mathbf{G}_k]), & \text{if } k = 1, \ldots, m-1 \\ \text{DECONV}_k(\mathbf{G}_k), & \text{if } k = m \end{cases}.$$

We compute a single value for each pixel by projecting the channel vector for each pixel using a single-layer perceptron with a ReLU non-linearity. Finally, we compute a probability distribution over the feature map using a SOFTMAX. The predicted location is the mode of the distribution.

6.2. Experimental Setup

The evaluation metrics are described in Section 3.2 and the data in Section 5.

Learning We use supervised learning. The gold label is a Gaussian smoothed distribution. The coordinate of the maximal value of the distribution is the exact coordinate where Touchdown is placed. We minimize the KL-divergence between the Gaussian and the predicted distribution.

Systems We evaluate three non-learning baselines: (a) RANDOM: predict a pixel at random; (b) CENTER: predict

Phenomenon	R2R		TOUCHDOWN						Example from TOUCHDOWN
			Overall		Navigation		SDR		
	c	μ	c	μ	c	μ	c	μ	
Reference to unique entity	25	3.7	25	10.7	25	9.2	25	3.2	... *You'll pass* ***three trashcans*** *on your left* ...
Coreference	8	0.5	22	2.4	15	1.1	22	1.5	... *a brownish colored brick building with a black fence around* ***it***...
Comparison	1	0.0	6	0.3	3	0.1	5	0.2	... *The bear is in the middle of the* ***closest*** *tire.*
Sequencing	4	0.2	22	1.9	21	1.6	9	0.4	... *Turn left at the* ***next*** *intersection* ...
Count	4	0.2	11	0.5	9	0.4	8	0.3	... *there are* ***two*** *tiny green signs you can see in the distance* ...
Allocentric spatial relation	5	0.2	25	2.9	17	1.2	25	2.2	... *There is a fire hydrant, the bear is* ***on top***
Egocentric spatial relation	20	1.2	25	4.0	23	3.6	19	1.1	... *up ahead there is some flag poles* ***on your right hand side***...
Imperative	25	4.0	25	5.3	25	5.2	4	0.2	... ***Enter*** *the next intersection and* ***stop*** ...
Direction	22	2.8	24	3.7	24	3.7	1	0.0	... *Turn* ***left****. Continue* ***forward*** ...
Temporal condition	7	0.4	21	1.9	21	1.9	2	0.1	... *Follow the road* ***until you see*** *a school on your right*...
State verification	2	0.1	21	1.8	18	1.5	16	0.8	... ***You should see*** *a small bridge ahead* ...

Table 3. Linguistic analysis of 25 randomly sampled development examples in TOUCHDOWN and R2R. We annotate each example for the presence and count of each phenomenon. We distinguish statistics for the entire text, navigation, and SDR segments in TOUCHDOWN. c is the number of instructions out of the 25 containing at least one example of the phenomenon; μ is the mean number of times each phenomenon appears in each of the 25 instructions.

the center pixel; (c) AVERAGE: predict the average pixel, computed over the training set. In addition to a two-level LINGUNET ($m = 2$), we evaluate three learning baselines: CONCAT, CONCATCONV, and TEXT2CONV. The first two compute a RESNET18 feature map representation of the image and then fuse it with the text representation to compute pixel probabilities. The third uses the text to compute kernels to convolve over the RESNET18 image representation. The Supplementary Material provides further details.

6.3. Results

Table 4 shows development and test results. The low performance of the non-learning baselines illustrates the challenge of the task. We also experiment with a UNET architecture that is similar to our LINGUNET but has no access to the language. This result illustrates that visual biases exist in the data, but only enable relatively low performance. All the learning systems outperform the non-learning baselines and the UNET, with LINGUNET performing best.

Figure 7 shows pixel-level predictions using LINGUNET. The distribution prediction is visualized as a heatmap overlaid on the image. LINGUNET often successfully solves descriptions anchored in objects that are unique in the image, such the *fire hydrant* at the top image. The lower example is more challenging. While the model correctly reasons that Touchdown is on *a light just above the doorway*, it fails to find the exact door. Instead, the probability distribution is shared between multiple similar locations, the space above three other doors in the image.

Method	A/C@40px	A/C@80px	A/C@120px	Dist
Development Results				
RANDOM	0.18 / 0.00	0.59 / 0.00	1.28 / 0.00	1185
CENTER	0.55 / 0.07	1.62 / 0.07	3.26 / 0.36	777
AVERAGE	1.88 / 0.07	4.22 / 0.29	7.14 / 0.79	762
UNET	10.86 / 2.69	13.94 / 3.31	16.69 / 3.91	957
CONCAT	13.70 / 3.22	17.85 / 4.46	21.16 / 5.47	917
CONCATCONV	13.56 / 3.24	18.00 / 4.58	21.42 / 5.71	918
TEXT2CONV	24.03 / 7.60	29.36 / 10.02	32.60 / 11.42	783
LINGUNET	24.81 / 7.73	32.83 / 13.00	36.44 / 15.01	729
Test Results				
RANDOM	0.21 / 0.00	0.78 / 0.00	1.89 / 0.00	1179
CENTER	0.31 / 0.00	1.61 / 0.21	3.93 / 0.57	759
AVERAGE	2.43 / 0.07	5.21 / 0.57	7.96 / 1.06	744
TEXT2CONV	24.82 / 8.21	30.40 / 11.73	34.13 / 13.32	747
LINGUNET	26.11 / 8.80	34.59 / 14.57	37.81 / 16.11	708

Table 4. Development and test results on the SDR task. We report accuracy/consistency (A/C) with different thresholds (40, 80, and 120) and mean distance error.

7. Navigation Baselines

7.1. Methods and Setup

We evaluate three non-learning baselines: (a) STOP: agent stops immediately; (b) RANDOM: Agent samples non-stop actions uniformly until reaching the action horizon; and (c) FREQUENT: agent always takes the most frequent action in the training set (`FORWARD`). We also evaluate two recent navigation models: (a) GA: gated-attention [6]; and (b) RCONCAT: a recently introduced model for landmark-based navigation in an environment that uses Street View images [24]. We represent the input images with RESNET18 features similar to the SDR task.

We use asynchronous training using multiple clients to generate rollouts on different partitions of the training data. We compute the gradients and updates using HOGWILD! [27] and ADAM learning rates [17]. We use supervised learning by maximizing the log-likelihood of actions in the reference demonstrations.

The details of the models, learning, and hyperparameters are provided in the Supplementary Material.

there will be a white/grey van parked on the right side of the road, and right behind the van on the walkway, there is a black fire hydrant with silver top, the touchdown is on the silver part of the fire hydrant.

a black doorway with red brick to the right of it, and green brick to the left of it. it has a light just above the doorway, and on that light is where you will find touchdown.

Figure 7. SDR pixel-level predictions with LINGUNET. Red-overlaid pixels indicate the Gaussian smoothed target location. Bright green overlay indicates the model's predicted probability distribution over pixels.

7.2. Results

Table 5 shows development and test results for our three valuation metrics (Section 3.1). The STOP, FREQUENT and RANDOM illustrate the complexity of the task. The learned baselines perform better. We observe that RCONCAT outperforms GA across all three metrics. In general though, the performance illustrates the challenge of the task. Appendix F includes additional navigation experiments, including single-modality baselines.

Method	TC	SPD	SED
Development Results			
STOP	0.0	26.7	0.0
FREQUENT	0.1	52.3	0.001
RANDOM	0.2	26.8	0.001
GA	7.9	21.5	0.077
RCONCAT	**9.8**	**19.1**	**0.094**
Test Results			
STOP	0.0	27.0	0.0
FREQUENT	0.0	53.1	0.0
RANDOM	0.2	26.9	0.001
GA	5.5	21.3	0.054
RCONCAT	**10.7**	**19.5**	**0.104**

Table 5. Development and test navigation results.

8. Complete Task Performance

We use a simple pipeline combination of the best models of the SDR and navigation tasks to complete the full task. Task completion is measured as finding Touchdown. We observe an accuracy of 4.5% for a threshold of 80px. In contrast, human performance is significantly higher. We estimate human performance using our annotation statistics [32]. To avoid spam and impossible examples, we consider only examples that were successfully validated. We then measure the performance of workers that completed over 30 tasks for these valid examples. This includes 55 workers. Because some examples required multiple tries to validate this set includes tasks that workers failed to execute but were later validated. The mean performance across this set of workers using the set of valid tasks is 92% accuracy.

9. Data Distribution and Licensing

We release the environment graph as panorama IDs and edges, scripts to download the RGB panoramas using the Google API, the collected data, and our code at touchdown.ai. These parts of the data are released with a CC-BY 4.0 license. Retention of downloaded panoramas should follow Google's policies. We also release RESNET18 image features of the RGB panoramas through a request form. The complete license is available with the data.

10. Conclusion

We introduce TOUCHDOWN, a dataset for natural language navigation and spatial reasoning using real-life visual observations. We define two tasks that require addressing a diverse set of reasoning and learning challenges. Our linguistically-driven analysis shows the data presents complex spatial reasoning challenges. This illustrates the benefit of using visual input that reflects the type of observations people see in their daily life, and demonstrates the effectiveness of our goal-driven data collection process.

Acknowledgements

This research was supported by a Google Faculty Award, NSF award CAREER-1750499, NSF Graduate Research Fellowship DGE-1650441, and the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program. We wish to thank Jason Baldridge for his extensive help and advice, and Valts Blukis and the anonymous reviewers for their helpful comments.

References

[1] P. Anderson, A. X. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir. On evaluation of embodied navigation agents. *CoRR*, abs/1807.06757, 2018.

[2] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. D. 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*, 2018.

[3] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. VQA: Visual question answering. In *IEEE International Conference on Computer Vision*, pages 2425–2433, 2015.

[4] Y. Bisk, D. Marcu, and W. Wong. Towards a Dataset for Human Computer Communication via Grounded Language Acquisition. In *Proceedings of the AAAI Workshop on Symbiotic Cognitive Systems*, 2016.

[5] 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.

[6] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov. Gated-Attention Architectures for Task-Oriented Language Grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[7] D. L. Chen and R. J. Mooney. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*, 2011.

[8] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft COCO Captions: Data Collection and Evaluation Server. *CoRR*, 2015.

[9] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[10] H. de Vries, K. Shuster, D. Batra, D. Parikh, J. Weston, and D. Kiela. Talk the Walk: Navigating New York City through Grounded Dialogue. *arXiv preprint arXiv:1807.03367*, 2018.

[11] O. Goldman, V. Latcinnik, E. Nave, A. Globerson, and J. Berant. Weakly supervised semantic parsing with abstract examples. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1809–1819, 2018.

[12] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi. IQA: Visual Question Answering in Interactive Environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[13] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9, 1997.

[16] S. Kazemzadeh, V. Ordonez, M. Matten, and T. Berg. ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 787–798, 2014.

[17] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations*, 2014.

[18] N. Kitaev and D. Klein. Where is Misty? Interpreting Spatial Descriptors by Modeling Regions in Space. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 157–166, 2017.

[19] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv preprint arXiv:1712.05474*, 2017.

[20] T.-Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*, 2014.

[21] M. MacMahon, B. Stankiewics, and B. Kuipers. Walk the Talk: Connecting Language, Knowledge, Action in Route Instructions. In *Proceedings of the National Conference on Artificial Intelligence*, 2006.

[22] J. Mao, J. Huang, A. Toshev, O. Camburu, A. Yuille, and K. Murphy. Generation and Comprehension of Unambiguous Object Descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11–20, 2016.

[23] C. Matuszek, N. FitzGerald, L. S. Zettlemoyer, L. Bo, and D. Fox. A Joint Model of Language and Perception for Grounded Attribute Learning. In *Proceedings of the International Conference on Machine Learning*, 2012.

[24] P. Mirowski, M. K. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, K. Kavukcuoglu, A. Zisserman, and R. Hadsell. Learning to Navigate in Cities without a Map. *Advances in Neural Information Processing Systems*, 2018.

[25] 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 *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018.

[26] D. Misra, J. Langford, and Y. Artzi. Mapping Instructions and Visual Observations to Actions with Reinforcement Learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1004–1015, 2017.

[27] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems*, 2011.

[28] S. Reed, Z. Akata, H. Lee, and B. Schiele. Learning Deep Representations of Fine-Grained Visual Descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–58, 2016.

[29] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR*, abs/1505.04597, 2015.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[31] A. Suhr, M. Lewis, J. Yeh, and Y. Artzi. A Corpus of Natural Language for Visual Reasoning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 217–223, 2017.

[32] A. Suhr, S. Zhou, I. F. Zhang, H. Bai, and Y. Artzi. A corpus for reasoning about natural language grounded in photographs. *CoRR*, abs/1811.00491, 2018.

[33] C. Yan, D. Misra, A. Bennnett, A. Walsman, Y. Bisk, and Y. Artzi. CHALET: Cornell House Agent Learning Environment. *arXiv preprint arXiv:1801.07357*, 2018.

[34] C. L. Zitnick and D. Parikh. Bringing Semantics into Focus Using Visual Abstraction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3009–3016, 2013.

A. Route Generation

We generate each route by randomly sampling two panoramas in the environment graph, and querying the Google Direction API[6] to obtain a route between them that follows correct road directions. Although the routes follow the direction of allowed traffic, the panoramas might still show moving against the traffic in two-way streets depending on which lane was used for the original panorama collection by Google. The route is segmented into multiple routes with length sampled uniformly between 35 and 45. We do not discard the suffix route segment, which may be shorter. Some final routes had gaps due to our use of the API. If the number of gaps is below three, we heuristically connect the detached parts of the route by adding intermediate panoramas, otherwise we remove the route segment. Each of the route segments is used in a separate instruction-writing task. Because panoramas and route segments are sampled randomly, the majority of route segments stop in the middle of a block, rather than at an intersection. This explicit design decision requires instruction-writers to describe exactly where in the block the follower should stop, which elicits references to a variety of object types, rather than simply referring to the location of an intersection.

B. Additional Data Analysis

We perform linguistically-driven analysis to two additional navigation datasets: SAIL [21, 7] and LANI [25], both using simulated environments. Both datasets include paragraphs segmented into single instructions. We performs our analysis at the paragraph level. We use the same categories as in Section 5. Table 6 shows the analysis results. In general, in addition to the more complex visual input, TOUCHDOWN displays similar or increased linguistic diversity compared to LANI and SAIL. LANI contains a similar amount of coreference, egocentric spatial relations, and temporal conditions, and more examples than TOUCHDOWN of imperatives and directions. SAIL contains a similar number of imperatives, and more examples of counts than TOUCHDOWN. We also visualize some of the common nouns and modifiers observed in our data (Figure 8).

C. SDR Pixel-level Predictions

Figures 9–14 show SDR pixel-level predictions for comparing the four models we used: LINGUNET, CONCAT, CONCATCONV, and CONCAT. Each figure shows the SDR description to resolve followed by the model outputs. We measure accuracy at a threshold of 80 pixels. Red-overlaid pixels visualize the Gaussian smoothed annotated target location. Green-overlaid pixels visualize the model's probability distribution over pixels.

[6] https://developers.google.com/maps/documentation/directions/start

D. SDR Experimental Setup Details

D.1. Models

We use learned word vectors of size 300. For all models, we use a single-layer, bi-directional recurrent neural network (RNN) with long short-term memory (LSTM) cells [15] to encode the description into a fixed-size vector representation. The hidden layer in the RNN has 600 unit. We compute the text embedding by averaging the RNN hidden states.

We provide the model with the complete panorama. We embed the panorama by slicing it into eight images, and projecting each image from a equirectangular projection to a perspective projection. Each of the eight projected images is of size 800×460. We pass each image separately through a RESNET18 [14] pretrained on ImageNet [30], and extract features from the fourth to last layer before classification; each slice's feature map is of size $128 \times 100 \times 58$. Finally, the features for the eight image slices are concatenated into a single tensor of size $128 \times 100 \times 464$.

CONCAT We concatenate the text representation along the channel dimension of the image feature map at each feature pixel and apply a multi-layer perceptron (MLP) over each pixel to obtain a real-value score for every pixel in the feature map. The multilayer perceptron includes two fully-connected layers with biases and ReLu non-linearities on the output of the first layer. The hidden size of each layer is 128. A SOFTMAX layer is applied to generate the final probability distribution over the feature pixels.

CONCATCONV The network structure is the same as CONCAT, except that after concatenating the text and image features and before applying the MLP, we mix the features across the feature map by applying a single convolution operation with a kernel of size 5×5 and padding of 2. This operation does not change the size of the image and text tensor. We use a the same MLP architecture as in CONCAT on the outputs of the convolution, and compute a distribution over pixels with a SOFTMAX.

TEXT2CONV Given the text representation and the featurized image, we use a kernel conditioned on the text to convolve over the image. The kernel is computed by projecting the text representation into a vector of size $409{,}600$ using a single learned layer without biases or non-linearities. This vector is reshaped into a kernel of size $5 \times 5 \times 128 \times 128$, and used to convolve over the image features, producing a tensor of the same size as the featurized image. We use a the same MLP architecture as in CONCAT on the outputs of this operation, and compute a distribution over pixels with a SOFTMAX.

LINGUNET We apply two convolutional layers to the image features to compute $\mathbf{F}_1$ and $\mathbf{F}_2$. Each uses a learned

Phenomenon	SAIL [21] Paragraphs		LANI [25] Paragraphs		TOUCHDOWN Overall		TOUCHDOWN Navigation		TOUCHDOWN SDR	
	c	μ	c	μ	c	μ	c	μ	c	μ
Reference to unique entity	24	4.0	25	7.2	25	10.7	25	9.2	25	3.2
Coreference	12	0.6	22	2.9	22	2.4	15	1.1	22	1.5
Comparison	0	0.0	2	0.1	6	0.3	3	0.1	5	0.2
Sequencing	4	0.2	2	0.1	22	1.9	21	1.6	9	0.4
Count	16	1.7	2	0.1	11	0.5	9	0.4	8	0.3
Allocentric spatial relation	9	0.4	3	0.2	25	2.9	17	1.2	25	2.2
Egocentric spatial relation	13	0.8	24	4.1	25	4.0	23	3.6	19	1.1
Imperative	23	4.5	25	9.0	25	5.3	25	5.2	4	0.2
Direction	23	4.5	25	5.8	24	3.7	24	3.7	1	0.0
Temporal condition	14	0.7	19	2.0	21	1.9	21	1.9	2	0.1
State verification	11	0.5	0	0.0	21	1.8	18	1.5	16	0.8

Table 6. Linguistic analysis of 25 randomly sampled development examples in TOUCHDOWN, SAIL, and LANI.

kernel of size 5×5 and padding of 2. We split the text representation into two vectors of size 300, and use two separate learned layers to transform each vector into another vector of size 16,384 that is reshaped to $1 \times 1 \times 128 \times 128$. The result of this operation on the first half of the text representation is $\mathbf{K}_1$, and on the second is $\mathbf{K}_2$. The layers do not contain biases or non-linearities. These two kernels are applied to $\mathbf{F}_1$ and $\mathbf{F}_2$ to compute $\mathbf{G}_1$ and $\mathbf{G}_2$. Finally, we use two deconvolution operations in sequence on $\mathbf{G}_1$ and $\mathbf{G}_2$ to compute $\mathbf{H}_1$ and $\mathbf{H_2}$ using learned kernels of size 5×5 and padding of 2.

D.2. Learning

We initialize parameters by sampling uniformly from $[-0.1, 0.1]$. During training, we apply dropout to the word embeddings with probability 0.5. We compute gradient updates using ADAM [17], and use a global learning rate of 0.0005 for LINGUNET, and 0.001 for all other models. We use early stopping with patience with a validation set containing 7% of the training data to compute accuracy at a threshold of 80 pixels after each epoch. We begin with a patience of 4, and when the accuracy on the validation set reaches a new maximum, patience resets to 4.

D.3. Evaluation

We compare the predicted location to the gold location by computing the location of the feature pixel corresponding to the gold location in the same scaling as the predicted probability distribution. We scale the accuracy threshold appropriately.

D.4. LINGUNET Architecture Clarifications

Our LINGUNET implementation for SDR task differs slightly from the original implementation [5]. We set the stride for both convolution and devonvolution operations to be 1, whereas in the original LINGUNET architecture the stride is set to 2. Experiments with the original implementation show equivalent performance.

E. Navigation Experimental Setup Details

E.1. Models

At each step, the agent observes the *agent context*. Formally, the agent context $\tilde{s}$ at time step t is a tuple $(\bar{x}_n, \mathbf{I}_t, \alpha_t, \langle(\mathbf{I}_1, \alpha_1, a_1), \ldots, (\mathbf{I}_{t-1}, \alpha_{t-1} a_{t-1})\rangle)$, where $\bar{x}_n$ is the navigation instruction, $\mathbf{I}_t$ is the panorama that is currently observed at heading α_t, and $\langle(\mathbf{I}_1, \alpha_1, a_1), \ldots, (\mathbf{I}_{t-1}, \alpha_{t-1} a_{t-1})\rangle$ is the sequence of previously observed panoramas, orientations, and selected actions. Given an agent context $\tilde{s}$, the navigation model computes action probabilities $P(a \mid \tilde{s})$.

We use learned word vectors of size 32 for all models. We map the instruction $\bar{x}_n$ to a vector $\mathbf{x}$ using a single-layer uni-directional RNN with LSTM cells with 256 hidden units. The instruction representation $\mathbf{x}$ is the hidden state of the final token in the instruction.

We generate RESNET18 features for each 360° panorama $\mathbf{I}_t$. We center the feature map according agent's heading α_t. We crop a $128 \times 100 \times 100$ sized feature map from the center. We pre-compute mean value along the channel dimension for every feature map and save the resulting 100×100 features. This pre-computation allows for faster learning. We use the saved features corresponding to $\mathbf{I}_t$ and the agent's heading α_t as $\hat{\mathbf{I}}_t$.

RCONCAT We modify the model of Mirowski *et al*. [24] for instruction-driven navigation. We use an RNN to embed the instruction instead of a goal embedding, and do not embed a reward signal. We apply a three-layer convolutional neural network to $\hat{\mathbf{I}}_t$. The first layer uses 32 8×8 kernels with stride 4, and the second layer uses 64 4×4 kernels with stride 4, applying ReLu non-linearities after each convolutional operation. We use a single fully-connected layer including biases of size 256 on the output of the convolutional operations to compute the observation's representation $\mathbf{I}'_t$. We learn embeddings $\mathbf{a}$ of size 16 for each action

Method	TC	SPD	SED
Development Results			
RCONCAT	**6.8**	23.4	**0.066**
GA	6.5	24.0	0.064
Test Results			
RCONCAT	**9.0**	22.6	**0.086**
GA + SUP	7.9	23.4	0.076

Table 7. Development and test navigation results using raw RGB images.

Method	TC	SPD	SED
Development Results			
RCONCAT NO-TEXT	24.48	7.26	0.07
RCONCAT NO-IMAGE	35.68	0.22	0.001
GA NO-TEXT	25.7	6.8	0.07
GA NO-IMAGE	50.1	0.1	0.0

Table 8. Single-modality development results.

a. For each time step t, we concatenate the instruction representation $\mathbf{x}$, observation representation $\mathbf{I}'_t$, and action embedding $\mathbf{a}_{t-1}$ into a vector $\tilde{\mathbf{s}}_t$. For the first time step, we use a learned embedding for the previous action. We use a single-layer RNN with 256 LSTM cells on the sequence of time steps. The input at time t is $\tilde{\mathbf{s}}_t$ and the hidden state is $\mathbf{h}_t$. We concatenate a learned time step embedding $\mathbf{t} \in \mathbb{R}^{32}$ with $\mathbf{h}_t$, and use a single-layer perceptron with biases and a SOFTMAX operation to compute $P(a_t|\tilde{s}_t)$.

GA We apply a three-layer convolutional neural network to $\hat{\mathbf{I}}_t$. The first layer uses 128 8×8 kernels with stride 4, and the second layer uses 64 4×4 kernels with stride 2, applying ReLu non-linearities after each convolutional operation. We use a single fully-connected layer including biases of size 64 on the output of the convolutional operations to compute the observation's representation $\mathbf{I}'_t$. We use a single hidden layer with biases followed by a sigmoid operation to map $\mathbf{x}$ into a vector $\mathbf{g} \in \mathbb{R}^{64}$. For each time step t, we apply a gated attention on $\mathbf{I}'_t$ using $\mathbf{g}$ along the channel dimension to generate a vector $\mathbf{u}_t$. We use a single fully-connected layer with biases and a ReLu non-linearity with $\mathbf{u}_t$ to compute a vector $\mathbf{v}_t \in \mathbb{R}^{256}$. We use a single-layer RNN with 256 LSTM cells on the sequence of time steps. The input at time t is $\mathbf{v}_t$ and the hidden state is $\mathbf{h}_t$. We concatenate a learned time step embedding $\mathbf{t} \in \mathbb{R}^{32}$ with $\mathbf{h}_t$, and use a single-layer perceptron with biases and a SOFTMAX operation to compute $P(a_t|\tilde{s}_t)$.

E.2. Learning

We train using asynchronous learning with six clients, each using a different split of the training data. We use supervised learning with HOGWILD! [27] and ADAM [17]. We generate a sequence of agent contexts and actions $\{(\tilde{s}_i, a_i)\}_{i=1}^N$ from the reference demonstrations, and maximize the log-likelihood objective:

$$\mathcal{J} = \max_\theta \sum_{i=1}^{N} \ln p_\theta(a_i \mid \tilde{s}_i) \ ,$$

where θ is the model parameters.

Hyperparameters We initialize parameters by sampling uniformly from $[-0.1, 0.1]$. We set the horizon to 55 during learning, and use an horizon of 50 during testing. We stop training using SPD performance on the development set. We use early stopping with patience, beginning with a patience value of 5 and resetting to 5 every time we observe a new minimum SPD error. The global learning rate is fixed at 0.00025.

F. Additional Navigation Experiments

F.1. Experiments with RGB Images

We also experiment with raw RGB images similar to Mirowski *et al*. [24]. We project and resize each 360° panorama $\mathbf{I}_t$ to a 60° perspective image $\hat{\mathbf{I}}_t$ of size $3\times84\times84$, where the center of the panorama is the agent's heading α_t. Table 7 shows the development and test results using RGB images. We observe better performance using RESNET18 features compared to RGB images.

F.2. Single-modality Experiments

We study the importance of each of the two modalities, language and vision, for the navigation task. We separately remove the embeddings of the language (NO-TEXT) and visual observations (NO-IMAGE) for both models. Table 8 shows the results. We observe no meaningful learning in the absence of the vision modality, whereas limited performance is possible without the natural language input. These results show both modalities are necessary, and also indicate that our navigation baselines (Table 5) benefit relatively little from the text.

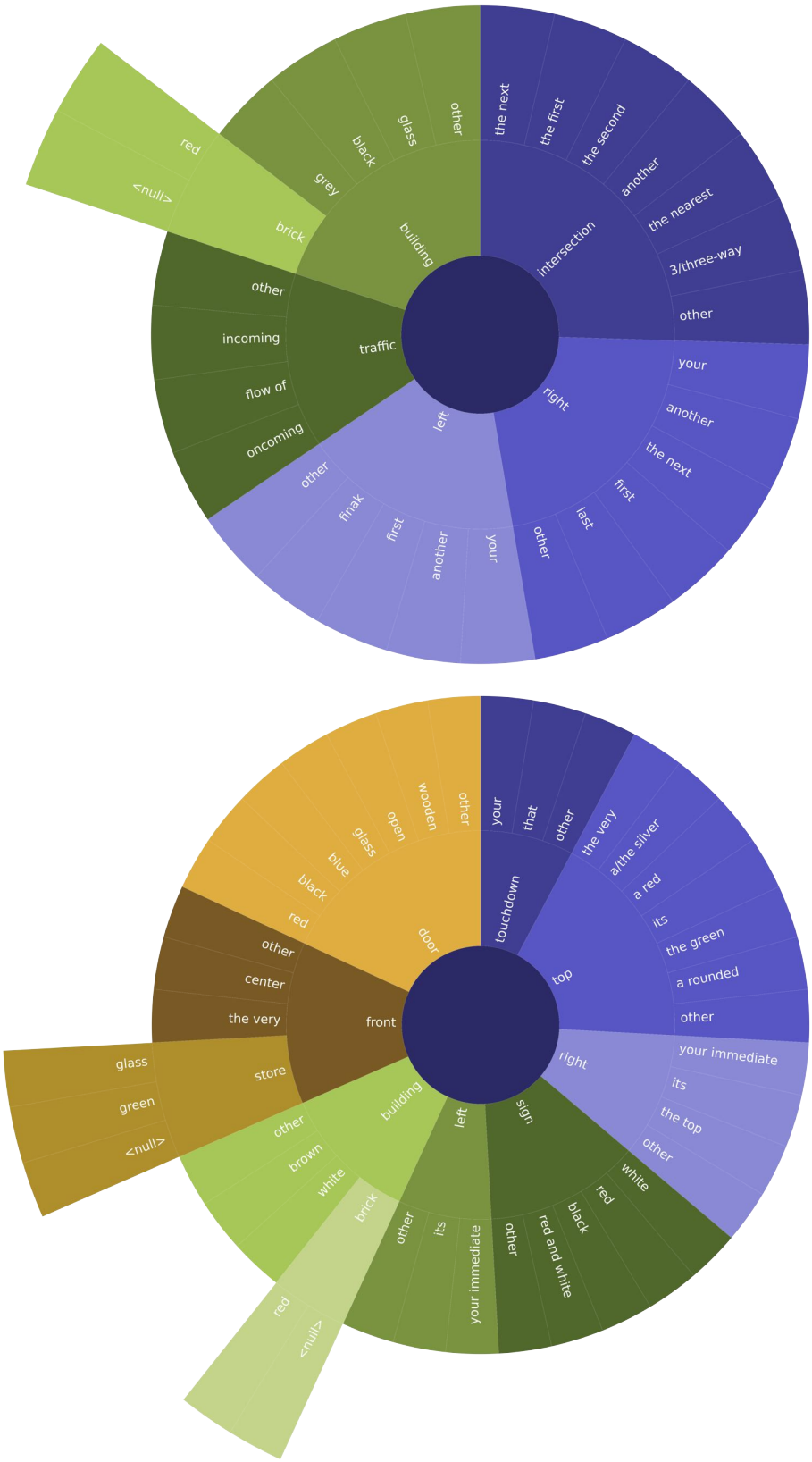
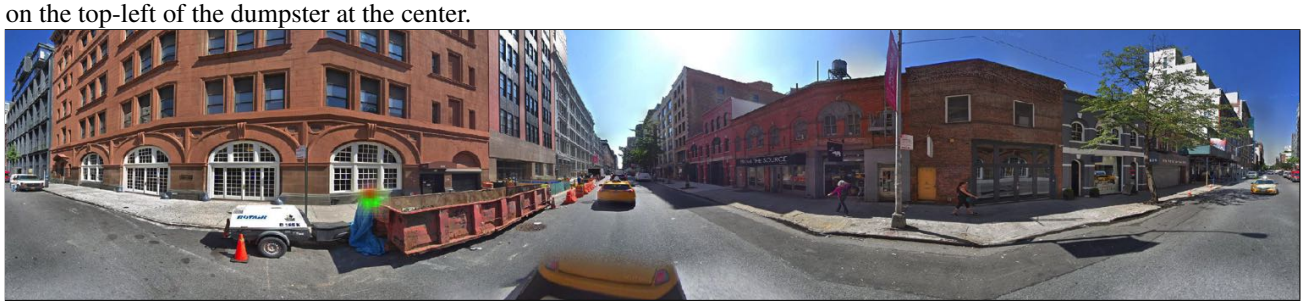


Figure 8. An illustration of the referential language in our navigation (top) and SDR (bottom) instructions. We ranked all nouns by frequency and removed stop words. We show the top five/eight nouns (most inner circle) for navigation and SDR. For each noun, we show the most common modifiers that prefix it. The size of each segment is not relative to the frequency in the data.

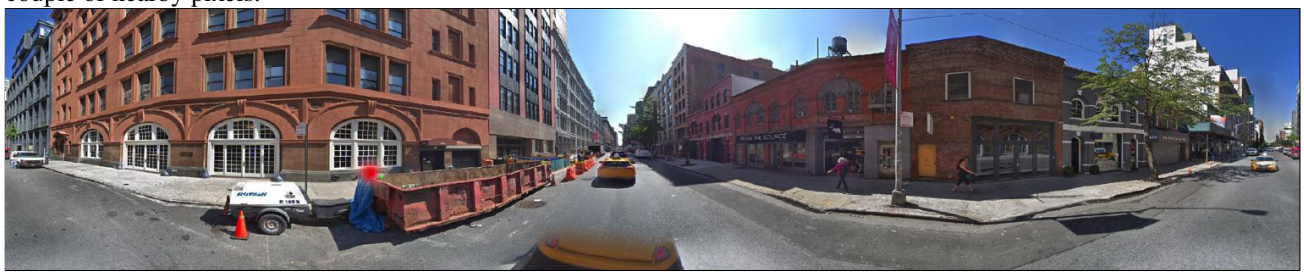
the dumpster has a blue tarp draped over the end closest to you. touchdown is on the top of the blue tarp on the dumpster.

LINGUNET The model correctly predicts the location of Touchdown, putting most of the predicted distribution (green) on the top-left of the dumpster at the center.



TEXT2CONV The model incorrectly predicts the location of Touchdown to the top of the car on the far right. While some of the probability mass is correctly placed on the dumpster, the pixel with the highest probability is on the car.

CONCATCONV The model correctly predicts the location of Touchdown. The distribution is heavily concentrated at a couple of nearby pixels.



CONCAT The prediction is similar to CONCATCONV.

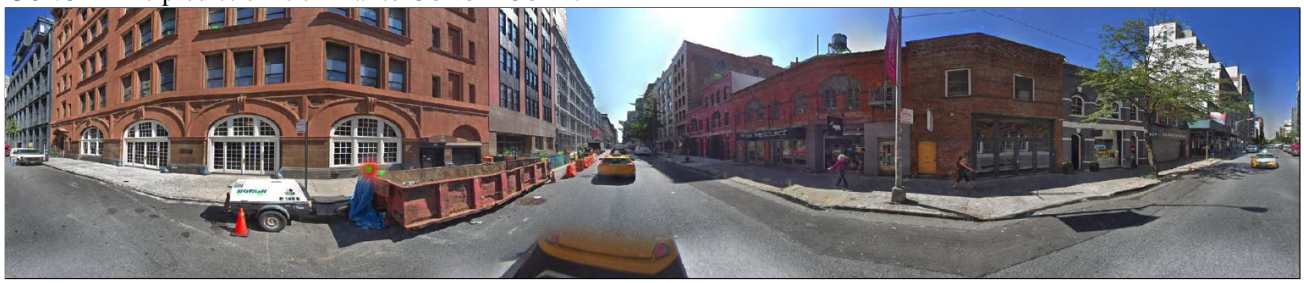


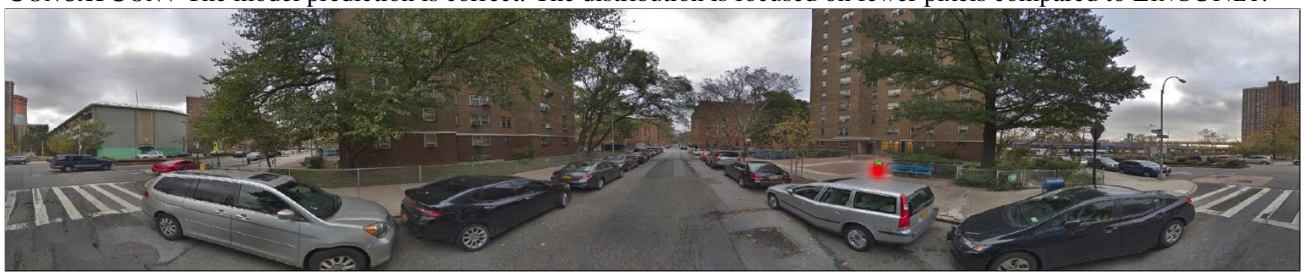
Figure 9. Three of the models are doing fairly well. Only TEXT2CONV fails to predict the location of Touchdown.

turn to your right and you will see a green trash barrel between the two blue benches on the right. click to the base of the green trash barrel to find touchdown.

LINGUNET The model accurately predicts the green trash barrel on the right as Touchdown's location.

TEXT2CONV The model predicts successfully as well. The distribution is focused on a smaller area compared to LINGUNET closer to the top of the object. This possibly shows a learned bias towards placing Touchdown on the top of objects that TEXT2CONV is more suceptible to.

CONCATCONV The model prediction is correct. The distribution is focused on fewer pixels compared to LINGUNET.



CONCAT The model prediction is correct. Similar to CONCATCONV, it focuses on a few pixels.

Figure 10. All the models predict the location of Touchdown correctly. Trash can is a relatively common object that workers use to place Touchdown in the dataset .

on your right is a parking garage, there is a red sign with bikes parked out in front of the garage, the bear is on the red sign.

LINGUNET The model predicted the location of Touchdown correctly to the red stop sign on the right side.

TEXT2CONV The model predicts the location of Touchdown correctly.

CONCATCONV The model predicts the location of Touchdown correctly.

CONCAT The model predicts the location of Touchdown correctly.

Figure 11. All the models predict the location of Touchdown correctly. Reference to a *red sign* are relatively common in the data (Figure 8) potentially simplifying this prediction.

touch down will be chillin in front of a sign on your right hand side about half way down this street,before you get to the sign there will be a multi color mural on the right w multiple colors and some writing on it.

LINGUNET The model fails to correctly predict the location of Touchdown, but is relatively close. The selected pixel is 104px from the correct one. The model focuses on the top of the sign instead of the bottom, potentially because of the more common reference to the top, which is visually distinguished.

TEXT2CONV The model fails to correctly predict the location of Touchdown, but is relatively close. The selected pixel is 96px from the correct one.

CONCATCONV The model fails to predict the location of Touchdown, instead focusing on a person walking on the left.

CONCAT The model fails to predict the location of Touchdown, instead of focusing the person walking on the left, the colorful sign mentioned in the description, and a car on the far right.

Figure 12. All the models fail to correctly identify the location of Touchdown. The predictions of LINGUNET, TEXT2CONV, and CONCATCONV seem to mix biases in the data with objects mentioned in the description, but fail to resolve the exact spatial description.

a row of blue bikes, touchdown is in the fifth bike seat in the row, from the way you came.

LINGUNET The model correctly identifies that a bike is mentioned, but fails to identify the exact bike or the location on the bike *seat*. Instead the the distribution is divided between multiple bikes.

TEXT2CONV Similar to LINGUNET, the model identifies the reference to *bikes*, but fails to identify the exact bike. The uncertainty of the model is potentially illustrated by how it distributes the probability mass.

CONCATCONV The model correctly predicts the location of Touchdown. While the distribution is spread across multiple bikes observed, the highest probability pixel is close enough (i.e., within 80 pixels) of the correct location.

CONCAT Similar to CONCATCONV, the model correctly predicts the location of Touchdown.

Figure 13. LINGUNET and TEXT2CONV fail to correctly identify the location, although their predicted distribution is focused on the correct set of objects. In contrast, the simpler models, CONCAT and CONCATCONV, correctly predict the location of Touchdown.

on your right is a parking garage, there is a red sign with bikes parked out in front of the garage, the bear is on the red sign.

LINGUNET The model misidentifies the red sign on the left hand side as the correct answer. It fails to resolve the spatial description, instead focusing on a more salient *red* object.

TEXT2CONV The model fails to predict the correct location, instead focusing on the red sign closer to the center.

CONCATCONV The model fails to predict the correct location, instead focusing on the red sign closer to the center.

CONCAT The model fails to predict the correct location, instead focusing on the red sign close to the center of the image.

Figure 14. All the models fail to identify the correct location. They focus unanimously on the red sign on the left hand side. They all ignore the reference to the *garage*, which is hard to resolve visually.