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.

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 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 $P(a_t | \tilde{s}_t)$.

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.

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.

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 $3 \times 84 \times 84$, 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.

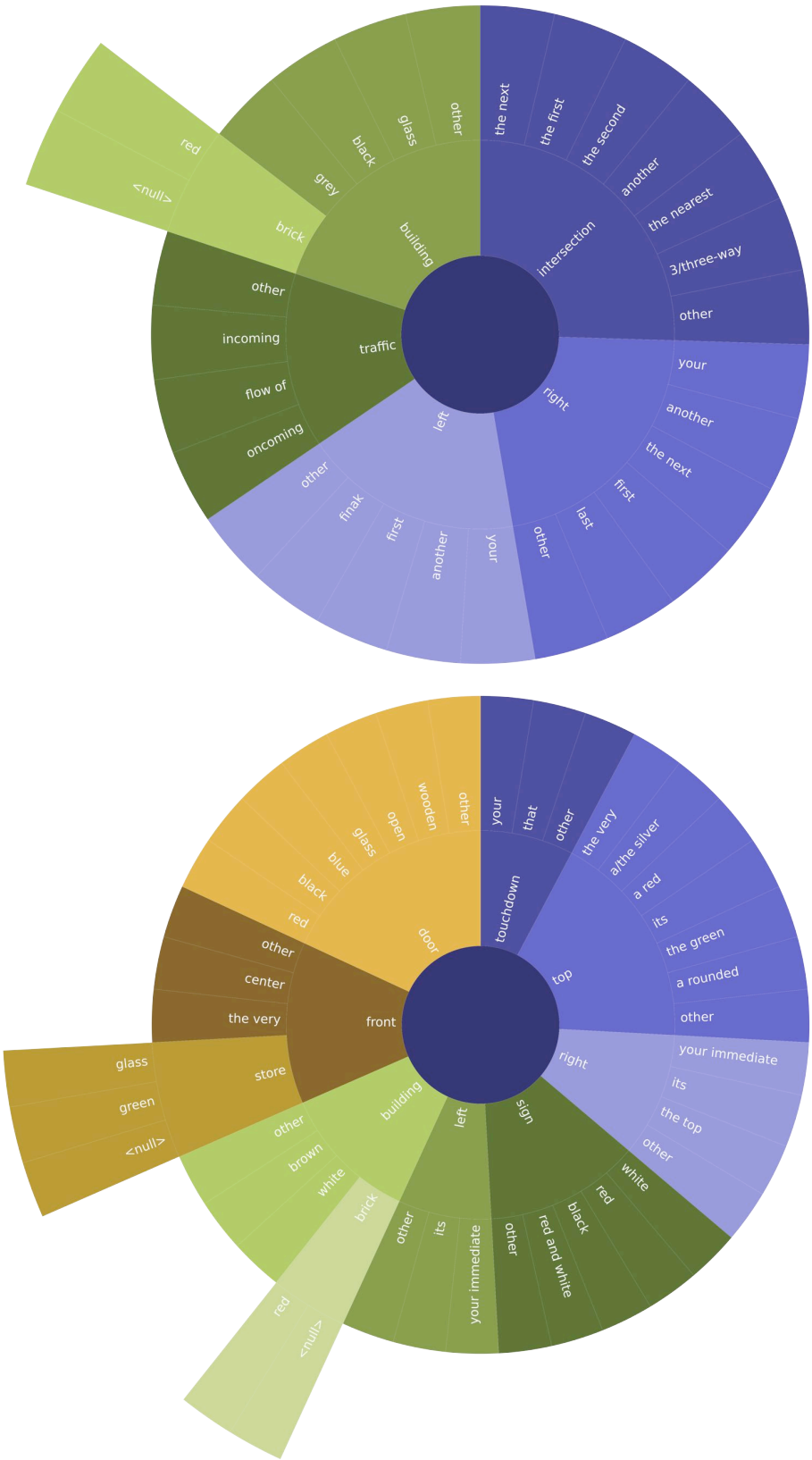
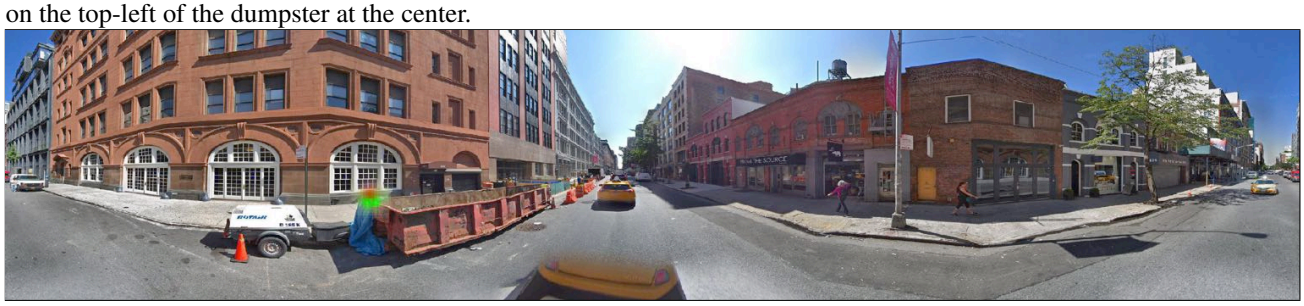


Figure 8. An illustration of the referential language in our navigation (top) and SDR (bottom) instructions. We ranked all nouns by frequency and removed stop words. We show the top five/eight nouns (most inner circle) for navigation and SDR. For each noun, we show the most common modifiers that prefix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

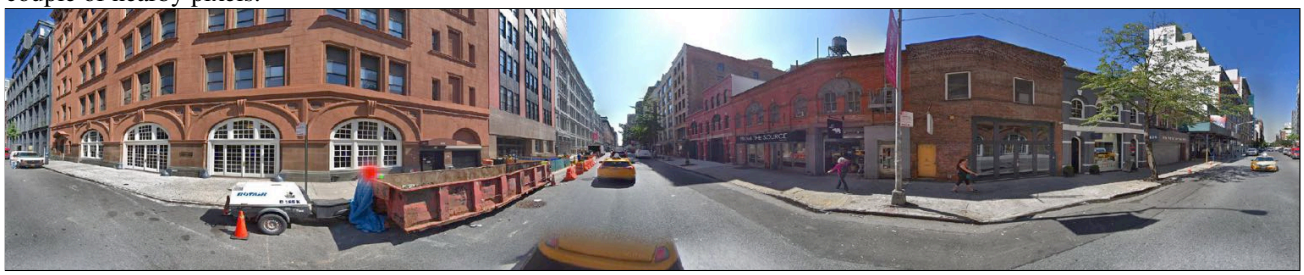
the dumpster has a blue tarp draped over the end closest to you. touchdown is on the top of the blue tarp on the dumpster.

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.

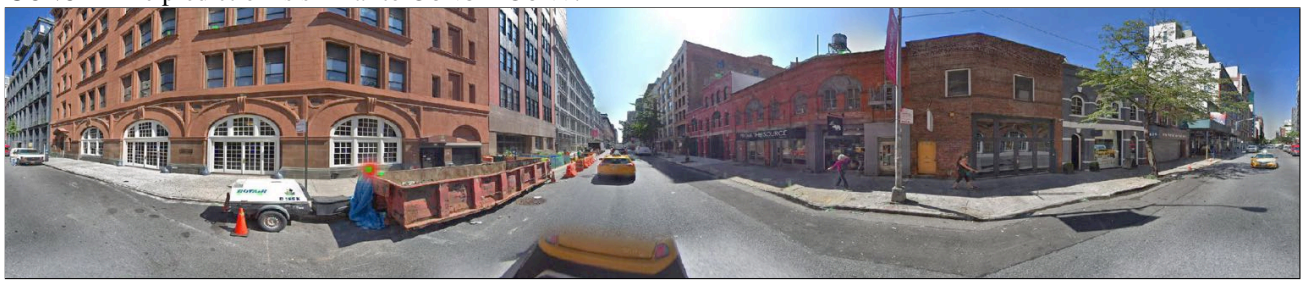


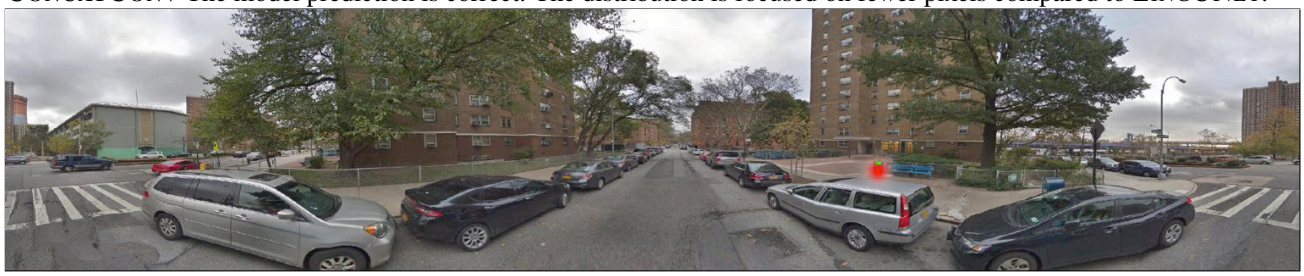
Figure 9. Three of the models are doing fairly well. Only TEXT2CONV fails to predict the location of Touchdown.

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.

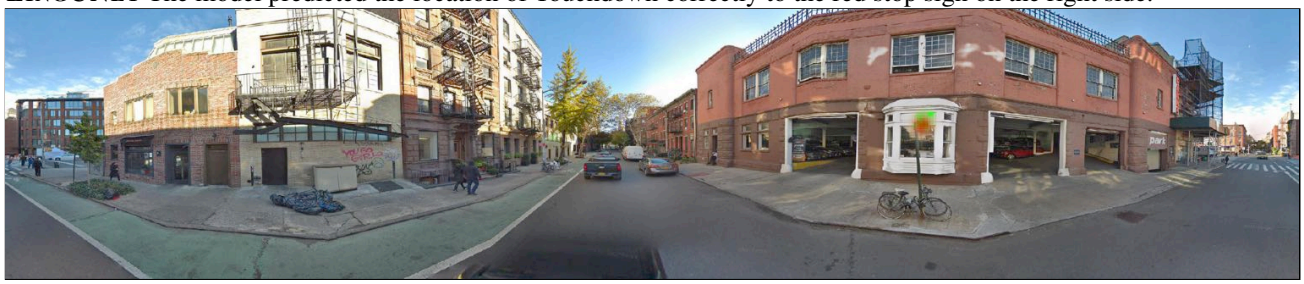


CONCAT The model prediction is correct. Similar to CONCATCONV, it focuses on a few pixels.

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.

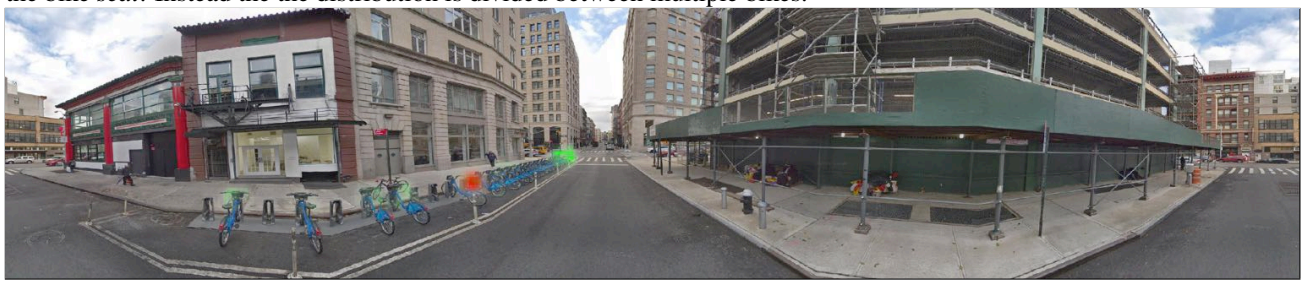
CONCATCONV The model fails to predict the location of Touchdown, instead focusing on a person walking on the left.

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.

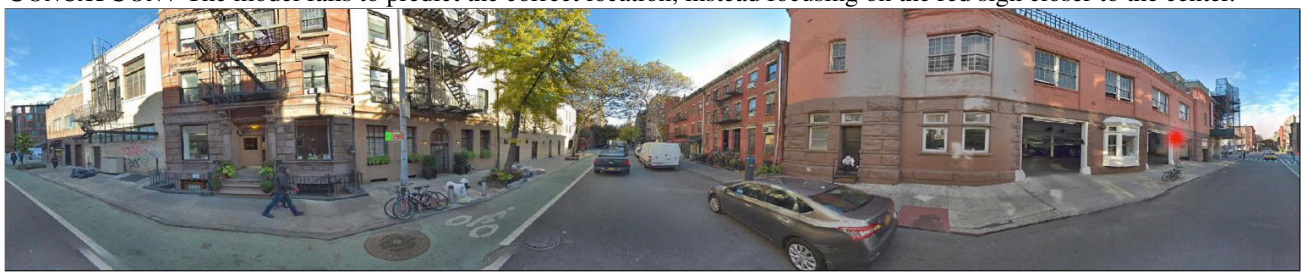
Figure 13. LINGUNET and TEXT2CONV fail to correctly identify the location, although their predicted distribution is focused on the correct set of objects. In contrast, the simpler models, CONCAT and CONCATCONV, correctly predict the location of Touchdown.

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.