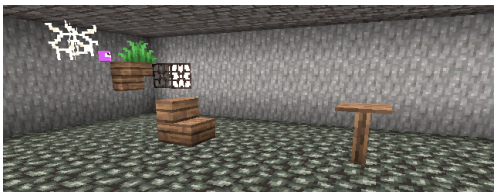
# Where is Misty? Interpreting Spatial Descriptors by Modeling Regions in Space

**Nikita Kitaev** and **Dan Klein**
Computer Science Division
University of California, Berkeley
{kitaev,klein}@cs.berkeley.edu

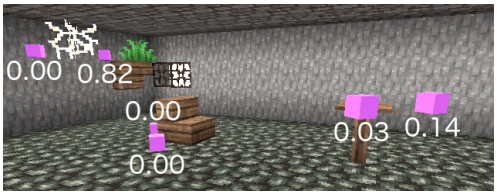
## Abstract

We present a model for locating regions in space based on natural language descriptions. Starting with a 3D scene and a sentence, our model is able to associate words in the sentence with regions in the scene, interpret relations such as *on top of* or *next to,* and finally locate the region described in the sentence. All components form a single neural network that is trained end-to-end without prior knowledge of object segmentation. To evaluate our model, we construct and release a new dataset consisting of Minecraft scenes with crowdsourced natural language descriptions. We achieve a 32% relative error reduction compared to a strong neural baseline.



*Misty is hanging in the air next to the wooden shelf with the plant on it.*

(a)




(b)

Figure 1: An example from our dataset. (a) The Minecraft scene and its natural language description. (b) Given the choice between six possible locations, our model assigns the highest probability to the location consistent with the natural language description.

## 1 Introduction

In this work, we present a model for grounding spatial descriptors in 3D scenes. Consider interpreting the instructions: *Take the book and put it on the shelf.* One critical element of being able to interpret this sentence is associating the referring expression *the book* with the corresponding object in the world. Another important component of understanding the command above is translating the phrase *on the shelf* to a location in space. We call such phrases spatial descriptors. While spatial descriptors are closely related to referring expressions, they are distinct in that they can refer to locations even when there is nothing there. An intuitive way to model this is to reason over spatial regions as first-class entities, rather than taking an object-centric approach.

Following a long tradition of using game environments for AI, we adopt Minecraft as the setting for our work. Minecraft has previously been used for work on planning and navigation (Oh et al., 2016; Tessler et al., 2016), and we expand on this by using it for grounded language understanding. As a sandbox game, it can be used to construct a wide variety of environments that capture many interesting aspects of the real world. At the same time, it is easy to extract machine-interpretable representations from the game.

We construct a dataset of Minecraft scenes with natural-language annotations, and propose a task that evaluates understanding spatial descriptors. Our task is formulated in terms of locating a pink, cube-shaped character named Misty given a scene, a natural language description, and a set of locations to choose from. An example from our dataset is shown in Figure 1. The Minecraft scene representation does not provide ground-truth information about object identity or segmentation, reflecting the fact that perceptual ambiguity is always



*Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 157–166
Copenhagen, Denmark, September 7–11, 2017. 

present in real-world scenarios. We do, however, assume the availability of 3D depth information (which, for real-world conditions, can be acquired using depth sensors such as RGBD cameras or LiDAR).

We propose and evaluate a neural network that combines convolutional layers operating over 3D regions in space with recurrent layers for processing language. Our model jointly learns to segment objects, associate them with words, and understand spatial relationships – all in an end-to-end manner. We compare with a strong neural baseline and demonstrate a relative error reduction of 32%.

The dataset and model described in this paper are available online.[1]

## 2 Related Work

Our task includes some of the same elements as referring-expression generation and interpretation. Past work on these tasks includes Golland et al. (2010), Krishnamurthy and Kollar (2013), Socher et al. (2014) and Kazemzadeh et al. (2014). A key difference is that spatial descriptors (as modeled in this paper) refer to locations in space, rather than to objects alone. For example, Krishnamurthy and Kollar (2013) convert natural language to a logical form that is matched against image segments, an approach that is only capable of reasoning about objects already present in the scene (and not skipped over by the segmentation process). Our model's ability to reason over spatial regions also differentiates it from past approaches to tasks beyond referring expressions, such as the work by Tellex et al. (2011) on natural-language commanding of robots. Recent work by Hu et al. (2016) on interpreting referring expressions can capture relationships between objects, relying on the construction of (subject, object, relation) tuples. Their model is limited in that it can only handle one such tuple per utterance. Our model does not have such a restriction, and it additionally expands to a 3D setting.

Our task is also related to work on Visual Question Answering, or VQA (Agrawal et al., 2015). While VQA uses free-form textual answers, our task places targeted emphasis on spatial reasoning by requiring outputs to be locations in the scene. Spatial reasoning remains an important capability for VQA systems, and is one of the elements featured in CLEVR (Johnson et al., 2016), a diagnostic dataset for VQA. Like in our dataset, visual percepts in CLEVR are based on machine-generated scenes. CLEVR also makes use of machine-generated language, while all language in our dataset is written by humans.

Another related task in NLP is spatial role labeling, which includes the identification of spatial descriptors and the assigning of roles to each of their constituent words. This task was studied by Kordjamshidi et al. (2011) and led to the creation of shared tasks such as SpaceEval (Pustejovsky et al., 2015). Our setting differs in that we consider grounded environments instead of studying text in isolation, and evaluate on task performance rather than logical correctness of interpretation.

Spatial descriptors are also present in the task of generating 3D scenes given natural language descriptions. Compared to a recent model by Chang et al. (2017) for scene generation, our model works with lower-level 3D percepts rather than libraries of segmented and tagged objects. We are also able to incorporate learning of vocabulary, perception, and linguistic structure into a single neural network that is trainable end-to-end.

## 3 Task

At its core, the ability to understand spatial descriptors can be formulated as mapping from a natural-language description to a particular location in space. In Figure 1, we show an instance of our task, which consists of the following components:

- $W$: a perceptual representation of the world
- $x$: the natural language description
- $\{y_1, y_2, \ldots, y_n\}$: the candidate set of locations that are under consideration
- $y^\star$: the true location that is being referred to in the scene

Given $W$ and $x$, a model must select which candidate location $y_i$ best matches the description $x$.

We will address the particulars of the above representation as we discuss the process for constructing our dataset. Each example $(W, x, \{y_1, \ldots, y_n\}, y^\star)$ in the dataset is made by generating a Minecraft scene (Section 3.1) and selecting a location as the target of description (Section 3.2). We then crowdsource natural language descriptions of the target location in space. To better anchor the language, we populate

[1] https://github.com/nikitakit/voxelworld

the target location with a cube-shaped character we name Misty, and ask workers to describe Misty's location (Section 3.3). We repeat this process for each example in the dataset.

### 3.1 Scene Generation and Representation

Each of our Minecraft scenes is set in a randomly-generated room. We select a random size for this room, and then populate it with a variety of objects. We include objects that can be placed on the floor (e.g. tables), mounted on the wall (e.g. torches), embedded in the wall (e.g. doors), or hanging from the ceiling (e.g. cobwebs).

We then discard ground-truth knowledge about object segmentation or identity in the process of saving our dataset. This allows our task to evaluate not only models' capacity for understanding language, but also their ability to integrate with perceptual systems. One way of approximating real-world observations would be to take a screenshot of the scene – however, a 2D projection does not provide all of the spatial information that a language user would reasonably have access to. We would like to use a 3D encoding instead, and Minecraft naturally offers a low-level (albeit low-resolution) voxel-based representation that we adopt for this work.

Each Minecraft world $W$ is encoded as a 3D grid of voxels, where a voxel may be empty or contain a particular type of "block," e.g. stone or wood. In general, what humans would interpret as single objects will be made of multiple Minecraft blocks – for example, the table in Figure 1 consists of a "wooden pressure plate" block on top of a "wooden fencepost" block. These same blocks can be used for other purposes as well: the "wooden fencepost" block is also part of fences, lamp-posts, and pillars, while the "wooden pressure plate" block can form shelves, countertops, as well as being placed on the ground to detect when something walks over it. We construct our Minecraft scenes specifically to include examples of such re-use, so that models capable of achieving high performance on this task must demonstrate the capacity to work without ground-truth segmentation or perfect object labeling.

The voxel-grid 3D representation is not specific to the virtual Minecraft setting: it is equally applicable to real-world data where depth information is available. The main difference is that each voxel would need to be associated with a feature vector rather than a block type. One use of such a representation is in Maturana and Scherer (2015)'s work on object classification from data collected with RGBD cameras and LiDAR, which uses a 3D convolutional neural network over a voxel grid. We do not explicitly handle occlusion in this work, but we imagine that real-world extensions can approach it using a combination of multi-viewpoint synthesis, occlusion-aware voxel embeddings, and restricting the set of voxels considered by the model.

### 3.2 Location Sampling

After constructing a scene with representation $W$, we proceed to sample a location $y^\star$ in the scene. Given our voxel-based scene representation, our location sampling is at voxel granularity. The candidate set we sample from, $\{y_1, \ldots, y_n\}$, consists of empty voxels in the scene. Locations that occur in the middle of a large section of empty space are hard to distinguish visually and to describe precisely, so we require that each candidate $y_i$ be adjacent to at least one object.

### 3.3 Natural Language Descriptions

For each scene-location pair $(W, y^\star)$ we crowdsource a natural language description $x$.

The choice of prompt for human annotators is important in eliciting good descriptions. At the location we are asking workers to refer to, we insert a pink-colored cube that we personify and name "Misty." We then ask workers to describe Misty's location such that someone can find her if she were to turn invisible. Having a visually salient target helps anchor human perception, which is why we chose a pink color that contrasts with other visual elements in the scene. We make sure to emphasize the name "Misty" in the instructions, which results in workers almost always referring to Misty by name or with the pronoun *she*. This avoids having to disambiguate a myriad of generic descriptions (*the pink block, the block, the target,* etc.) for what is fundamentally an artificial construct.

To make sure that humans understand the 3D structure of the scene as they describe it, we give them access to a 3D view of the environment and require that they move the camera before submitting a description. This helped increase the quality of our data.

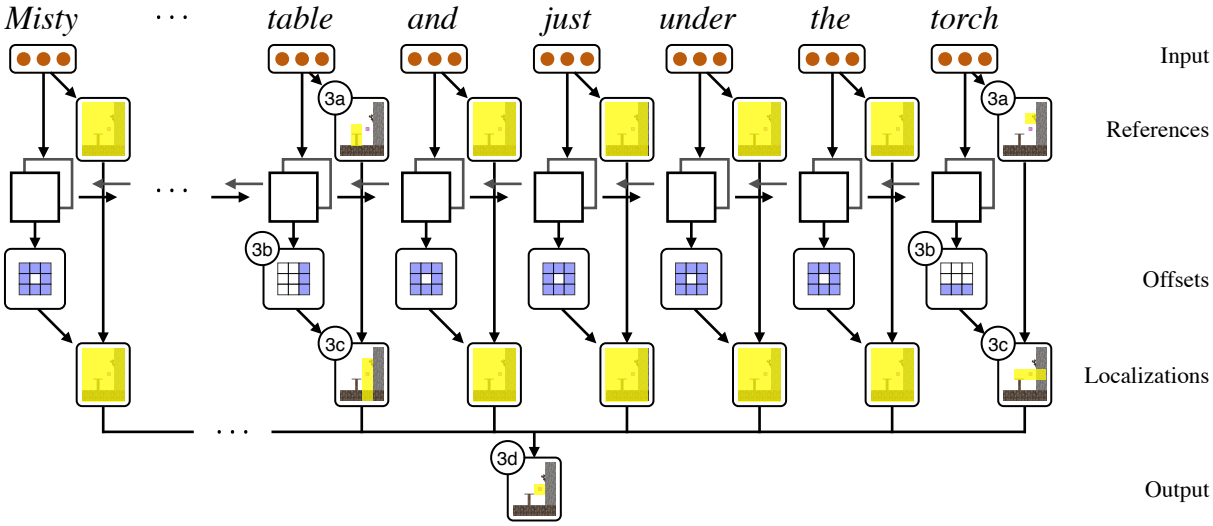



Figure 2: Our model architecture. Note that while the schematic illustrations are shown in 2D, our actual model operates in 3D. Zoomed-in versions of the marked references, offsets, localizations, and output are shown in Figure 3.

## 4 Model

We next present our model for this task. Our model architecture is shown in Figure 2, with some of the quantities it operates over highlighted in Figure 3. Throughout this section, we will use the example description *Misty is to the right of the table and just under the torch.* Note that while the accompanying scene illustrations are shown in 2D for visual clarity, our actual model operates in 3D and on larger scene sizes.

Our model first associates words with regions in the world. There is no notion of object segmentation in the dataset, so the references it produces are just activations over space given a word. Activations are computed for all words in the sentence, though they will only be meaningful for words such as *table* and *torch* (Figure 3a). Our model next determines the spatial relationships between referenced objects and Misty, using information provided by context words such as *right* and *under*. These relationships are represented as 3D convolutional offset filters (Figure 3b). For each word, its reference and offset filter are convolved to get a localization, i.e. an estimate of Misty's location (Figure 3c). Finally, our model aggregates localizations across all words in the sentence, combining the information provided by the phrases *to the right of the table* and *just under the torch* (Figure 3e).

The following sections describe in more detail how references (Section 4.1), offsets (Section 4.2), and localizations (Section 4.3) are computed.

### 4.1 Input and References

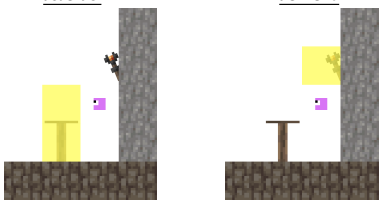
The first component of our model is responsible for associating words with the voxels that they refer to. It assigns a real-valued score $s(x_t, y)$ to each pair consisting of word $x_t$ and voxel coordinate $y$.

High scores correspond to high compatibility; for any given word, we can visualize the set $s(x_t, \cdot)$ of scores assigned to different voxels by interpreting it as logits that encode a probability distribution over blocks in the scene. In the example, the word *table* would ideally be matched to the uniform reference distribution over blocks that are part of a table, and similarly for the word *torch* (Figure 3a).

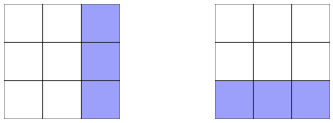
The word-voxel scores are computed by combining word and block embeddings. To take advantage of additional unsupervised language and world data, we start with pretrained word embeddings and context-aware location embeddings $f(W, y)$. The function $f$ consists of the first two layers of a convolutional neural network that is pretrained on the task of predicting a voxel's identity given the 5x5x5 neighborhood around it. Since $f$ fails to take into account the actual voxel's identity, we add additional embeddings $V$ that only consider single blocks. The score is then computed as $s(x_t, y) = w_t^\top A f(W, y) + w_t^\top v_y$, where $w_t$ is the word embedding and $v_y$ is the single-block embedding. The parameter matrix

*Misty is to the right of the table and just under the torch.*

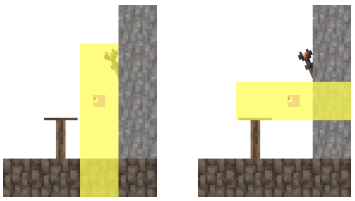
*table* *torch*



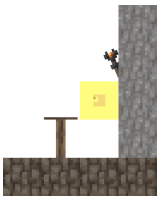
(a) Reference distributions for the words *table* and *torch*.



(b) Offset distributions that will be applied to the references for *table* and *torch*. These are calculated based on the language context, including the words *right* and *under*.



(c) Localizations for Misty, given the words *table* and *torch* in context.



(d) Output

(e) Output distribution produced by intersecting the localizations for each word.

Figure 3: A schematic 2D depiction of the representations used throughout our neural network, zoomed in from Figure 2. Our model (a) matches words with objects in the scene, (b) determines offsets from landmark objects to Misty's location, (c) combines these pieces of information to form per-word localizations of Misty, and then (d) uses all localizations to guess Misty's location.

$A$ and the single-block embedding matrix $V$ are trained end-to-end with the rest of the model.

References are computed for all words in the sentence – including function words like *to* or *the*. To signify that a word does not refer to any objects in the world, the next layer of the network expects that we output a uniform distribution over all voxels. Outputting uniform distributions also serves as a good initialization for our model, so we set the elements of $A$ and $V$ to zero at the start of training (our pretrained word embeddings are sufficient to break symmetry).

### 4.2 Offsets

The per-word references described in Section 4.1 do not themselves indicate Misty's location. Rather, they are used in a spatial descriptor like *to the right of the table*. For every word, our model outputs a distribution over offset vectors that is used to redistribute scores from object locations to possible locations for Misty (Figure 3b). For example, if probability mass is placed on the "one-block-to-the-right" offset vector, this corresponds to predicting that Misty will be one block to the right of the voxels that a word refers to. Offset scores $o_t$ are assigned based on the context the word $x_t$ occurs in, which allows the model to incorporate information from words such as *right* or *under* in its decisions. This is accomplished by running a bidirectional LSTM over the embeddings $w_t$ of the words in the sentence, and using its output to compute offset probabilities:

$$[z_0, z_1, \ldots] = \text{BiLSTM}([w_0, w_1, \ldots])$$
$$o'_t = M z_t$$
$$o_t(i) \propto \exp\left(o'_t(i)\right)$$

Each set of offset scores $o_t$ is reshaped into a 3x3x3 convolutional filter, except that we structurally disallow assigning any probability to the no-offset vector in the center. As a parameter-tying technique, the trainable matrix $M$ is not full-rank; we instead decompose it such that the log-probability of an offset vector factors additively over the components in a cylindrical coordinate system.

### 4.3 Localizations and Output

For each word, the 3D tensor of word-voxel scores $s(x_t, \cdot)$ is convolved with the offset distribution $o_t$ to produce a distribution of localizations for Misty, $d_t(y)$. A 2D illustration of the result is shown in Figure 3c. Localizations are then summed across all words in the sentence, resulting in a single score for each voxel in the scene (Figure 3e). These scores are interpreted as logits corresponding to a probability distribution over possible locations for Misty:

$$d_t(y) = s(x_t, y) * o_t$$
$$p(y) \propto \exp\left\{\sum_t d_t(y)\right\}$$

Not all words will have localizations that provide information about Misty – for some words

the localizations will just be a uniform distribution. We will refer to words that have low-entropy localizations as *landmarks*, with the understanding that being a landmark is actually a soft notion in our model.

Our offset filters $o_t$ are much smaller than our voxel grid, which means that convolving any offset filter with a uniform reference distribution over the voxel grid will also result in a uniform localization distribution (edge effects are immaterial given the small filter size and the fact that Misty is generally not at the immediate edges of the scene). Conversely, given non-uniform references almost any set of offsets will result in a non-uniform localization. The architecture for computing references can output uniform references for function words (like *to* or *the*), but it lacks the linguistic context to determine when words refer to objects but should not be interpreted as landmarks (e.g. when they are part of exposition or a negated expression). We therefore include an additional not-a-landmark class that is softmax-normalized jointly with the offset vector distribution $o_t$. Probability assigned to this class subtracts from the probability mass for the true offset directions (and therefore from the localizations) – if this class receives a probability of 1, the corresponding localizations will not contribute to the model output.

### 4.4 Loss and Training

We use a softmax cross-entropy loss for training our model. During training, we find that it helps to not use the candidate set $\{y_1, y_2, \ldots, y_n\}$ and instead calculate a probability $p(y)$ for all blocks in the scene, including solid blocks that cannot possibly contain Misty (perhaps because this penalizes inferring nonsensical spatial relationships).

We run the Adam optimizer (Kingma and Ba, 2014) with step size 0.001 for 100 epochs using batch size 10. We keep an exponential moving average of our trainable parameters, which we save every two epochs. We then select the saved model that has the highest performance on our development set.

We perform several regularization and data augmentation techniques in order to achieve better generalization. Each time we sample a training example, we select a random 19x19x19 crop from the full scene (as long as Misty's location is not cropped out). We also disallow using the context-based block embeddings for the first 20 epochs by holding the parameter matrix $A$ described in Section 4.1 fixed at zero, forcing the model to first learn to associate vocabulary with local features and only later expand to capture the compositional aspects of the environment.

For the natural language descriptions, all tokens are converted to lowercase as part of preprocessing. During training we apply word-level dropout (i.e. replacing words with an UNK token) in the LSTM responsible for computing offsets.

## 5 Evaluation

### 5.1 Evaluation Metric

In evaluating this task, we would like to use a metric that can provide meaningful comparison of our model with baseline and human performance. The set of all possible locations for Misty is large enough that it is hard even for a human to guess the correct block on the first try, especially when some descriptions are only precise to within 1 or 2 blocks. The size of this set also varies from scene to scene.

Therefore for our evaluation, we restrict the set $\{y_1, \ldots, y_n\}$ to 6 possible locations: Misty's true location and 5 distractors. This represents a less ambiguous problem that is much easier for humans, while also allowing for the evaluation of future models that may require an expensive computation for each candidate location considered. Our procedure for selecting the distractors is designed to ensure that we test both local and global scene understanding. Each set of six choices is constructed to consist of three clusters of two candidates each. Each cluster location is anchored to a landmark – we sample a landmark block adjacent to Misty and two additional landmark blocks from the entire scene, such that the pairwise distances between landmarks are at least 4 units. We then sample one distractor near Misty's landmark and two distractors near both of the other landmarks.

### 5.2 Dataset

To make our development and test sets, we construct this six-option variation from a subset of our collected data. For each such example we crowdsource two human solutions using Mechanical Turk. Examples where both humans answered correctly are partitioned into a development and a test set. This filtering procedure serves as our primary method of excluding confusing or uninformative descriptions from the evaluation con-

| | |
|---|---|
| (a) | *When you come in the door, she's on the floor to the right, just in front of the flower.* |
| (b) | *Misty is facing to the right of the brown door.* |
| (c) | *If you were to walk through the door that is on the same wall as the table and plank of floating wood, Misty would be to the left of the door. She is eye level with the plank of wood and floating in front of it.* |
| (d) | *Misty is in the ground and she is front of door.* |
| (e) | *Misty is located under a table that is connected to the wall. She is at ground level.* |

Table 1: Five natural-language descriptions sampled at random from our dataset.

ditions. We also collect a third human solution to each example in the development and test sets to get an independent estimate of human performance on our task. The final dataset consists of 2321 training examples, 120 dev set examples, and 200 test set examples.

The natural-language descriptions across the full dataset use a vocabulary of 1015 distinct tokens (case-insensitive but including punctuation). The average description length is 19.02 tokens, with a standard deviation of 10.00 tokens. The large spread partially reflects the fact that some people gave short descriptions that referenced a few landmarks, while others gave sequences of instructions on how to find Misty. As a point of comparison, the ReferIt dataset (Kazemzadeh et al., 2014) has a larger vocabulary of 9124 tokens, but a shorter average description length of 3.52 tokens (with a standard deviation of 2.67 tokens).

A random sampling of descriptions from our dataset is shown in Table 1.

### 5.3 Quantitative Results

Quantitative results are shown in Table 2. Our evaluation metric is constructed such that there is an easily interpretable random baseline. We also evaluate a strong neural baseline that uses an approach we call Seq2Emb. This baseline converts the sentence into a vector using a bidirectional LSTM encoder, and also assigns vector embeddings to each voxel using a two-layer convolutional neural network. The voxel with an embedding that most closely matches the sentence embedding is chosen as the answer.

Our model achieves noticeable gains over the baseline approaches. At the same time, there remains a gap between our model and individual human performance. We see this as an indication that we have constructed a task with appropriate difficulty: it is approachable by building on the current state-of-the-art in machine learning and NLP, while presenting challenges that can motivate continued work on understanding language and how it relates to descriptions of the world.

| | Dev Set | Test Set |
|---|---|---|
| Random Baseline | 16.67 | 16.67 |
| Seq2Emb Baseline | 52.50 | 44.50 |
| Our Model | 67.50 | 62.50 |
| Human | 85.83 | 87.50 |

Table 2: Success rates for our dataset split. Our model is able to outperform a strong neural baseline (Seq2Emb).

| | % correct |
|---|---|
| Full model | **67.5** |
| −contextual block embeddings | 65.0 |
| −LSTM (use 3-word convolutions instead) | 62.5 |
| −language-dependent spatial operators | 61.7 |

Table 3: Development set results for our full model and three independent ablations.

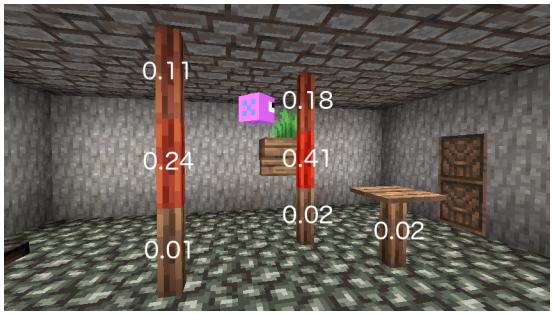
### 5.4 Ablation Study

We next conduct an ablation study to evaluate the contribution of the individual elements in our model. Our ablation results on the development set are shown in Table 3.

In our first ablation, we remove the compositional block embeddings that make use of multiple blocks. The resulting performance drop of 2.5% reflects the fact that our model uses multi-block information to match words with objects.

We next replace the LSTM in our full model with a 3-word-wide convolutional layer. A single word of left- and right-context provides limited ability to incorporate spatial descriptor words like *left* and *right*, or to distinguish landmarks used to locate Misty from words providing exposition about the scene. This ablation solves 5% fewer examples than our full model, reflecting our LSTM's ability to capture such phenomena.

Finally, we try holding the distribution over offset vectors fixed, by making it a trainable variable rather than a function of the language. This corresponds to enforcing the use of only one spatial




*Misty is floating in the middle of the room. She in the upper half of the room, between the two* ***poles****.*

Figure 4: Reference distribution representing our model's belief of which blocks the word *poles* refers to. Our model assigns the majority of the probability mass to the poles, while ignoring a table leg that is made of the same block type. Note that the seven numbers overlaid on top account for more than 99% of the total probability mass, and that each of the remaining blocks in the scene has a probability of at most 0.025%.

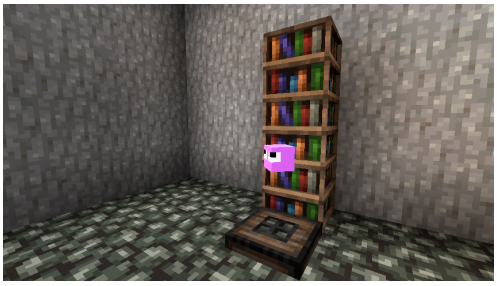
operator that roughly means 'near.' We retain the LSTM for the sole purpose of assigning a score to the not-a-landmark class, meaning that contextual information is still incorporated in the decision of whether to classify a word as a landmark or not. The resulting accuracy is 5.8% lower than our full model, which makes this the worst-performing of our ablations. These results suggest that the ability to infer spatial directions is important to our model's overall performance.

### 5.5 Qualitative Examination

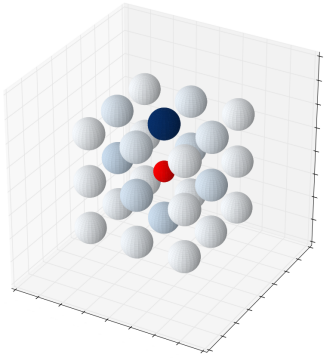
The modular design of our model allows us to examine the individual behavior of each component in the network, which we explore in this section.

We find that our algorithm is able to learn to associate words with the corresponding voxels in the world. Figure 4 shows the reference distribution associated with the word *poles*, which is constructed by applying a softmax operation to the word-voxel scores for that word. Our algorithm is able to correctly segment out the voxels that are a part of the pole. Moreover, the table on the right side of the scene has a table leg made of the same block type as the pole – and yet, it is is assigned a low probability. This shows that our model is capable of representing compositional objects, and can learn to do so in an end-to-end manner.

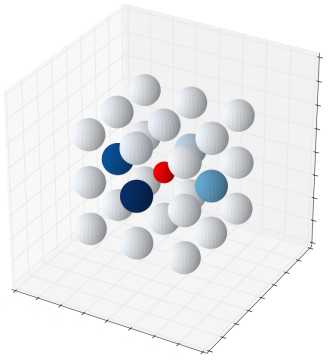
We next examine the offset distributions computed by our model. Consider the scene and description shown in Figure 5a. The offset vector distribution at the word *platform*, shown in Figure 5b, shows that the model assigns high proba-



(a) *To the left of the room, there is a bookcase with a platform directly in front of it. Misty is right above the* ***platform****.*

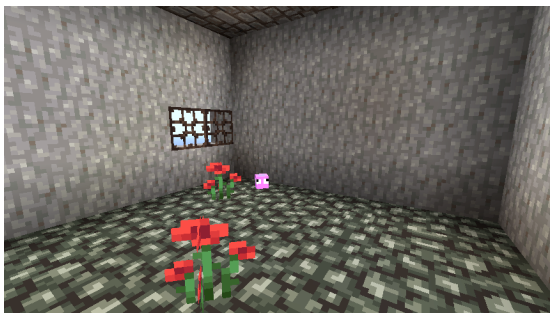


(b) *Misty is right above the platform.*



(c) *Misty is in front of the platform.*

Figure 5: Effects of language context on offset vector distributions. In (a), we show the scene and its description. In (b), we visualize the offset vector distribution at the word *platform*, i.e. the 3D convolutional filter that is applied after finding the platform location. The red dot that indicates the center of the filter will be matched with the platform location. In (c), we have artificially replaced the words *right above* with *in front of*, resulting in a substantial change to this distribution.



*Misty is between the wall and the flowers that are close to the corner.*

Figure 6: Our algorithm interprets this sentence as *Misty is near the wall and the flowers and close to the corner*. This intersective interpretation is sufficient to correctly guess Misty's location in this scene (as well as others in the dataset).

bility to Misty being above the platform. In Figure 5c, we show the effects of replacing the phrase *right above* with the words *in front of*. This example illustrates our model's capacity for learning spatial directions. We note that the offset distribution given the phrase *in front of* is not as peaked as it is for *right above*, and that distributions for descriptions saying *left* or *right* are even less peaked (and are mostly uniform on the horizontal plane). One explanation for this is the ambiguity between speaker-centric and object-centric reference frames. The reference frame of our convolutional filters is the same as the initial camera frame for our our annotators, but this may not be the true speaker-centric frame because we mandate that annotators move the camera before submitting a description.

We next highlight our model's ability to incorporate multiple landmarks in making its decisions. Consider the scene and description shown in Figure 6. The room has four walls, two flowers, and four corners – no single landmark is sufficient to correctly guess Misty's location. Our model is able to localize the flowers, walls, and corners in this scene and intersect them to locate Misty. Strictly speaking, this approach is not logically equivalent to applying a two-argument *between* operator and recognizing the role of *that* as a relativizer. This is a limitation of our specific model, but the general approach of manipulating spatial region masks need not be constrained in this way. It would be possible to introduce operations into the neural network to model recursive structure in the language. In practice, however, we find that the intersective interpretation suffices for many of the descriptions that occur in our dataset.

## 6 Conclusion

In this paper, we define the task of interpreting spatial descriptors, construct a new dataset based on Minecraft, and propose a model for this task. We show that convolutional neural networks can be used to reason about regions in space as first-class entities. This approach is trainable end-to-end while also having interpretable values at the intermediate stages of the neural network.

Our architecture handles many of the linguistic phenomena needed to solve this task, including object references and spatial regions. However, there is more work to be done before we can say that the network completely understands the sentences that it reads. Our dataset can be used to investigate future models that expand to handle relativization and other recursive phenomena in language.

## Acknowledgments

We thank the many workers on Mechanical Turk who contributed to the creation of our dataset.

This work was made possible by the open source tooling developed around and inspired by Minecraft; in particular we would like to thank the developers of the `voxel.js` project and associated plugins, as well as the developers of `mcedit2`.

Nikita Kitaev is supported by an NSF Graduate Research Fellowship. This research was supported by DARPA through the XAI program.

## References

Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, and Devi Parikh. 2015. VQA: Visual Question Answering. *arXiv:1505.00468 [cs]*.

Angel X. Chang, Mihail Eric, Manolis Savva, and Christopher D. Manning. 2017. SceneSeer: 3d Scene Design with Natural Language. *arXiv:1703.00050 [cs]*.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *EMNLP*, pages 410–419.

Ronghang Hu, Marcus Rohrbach, Jacob Andreas, Trevor Darrell, and Kate Saenko. 2016. Modeling Relationships in Referential Expressions with Compositional Modular Networks. *arXiv:1611.09978 [cs]*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross

Girshick. 2016. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. *arXiv:1612.06890 [cs]*.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L. Berg. 2014. ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *EMNLP*, pages 787–798.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*.

Parisa Kordjamshidi, Martijn Van Otterlo, and Marie-Francine Moens. 2011. Spatial role labeling: Towards extraction of spatial relations from natural language. *ACM Transactions on Speech and Language Processing (TSLP)*, 8(3):4.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1:193–206.

Daniel Maturana and Sebastian Scherer. 2015. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE.

Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, and Honglak Lee. 2016. Control of Memory, Active Perception, and Action in Minecraft. *arXiv:1605.09128 [cs]*.

James Pustejovsky, Parisa Kordjamshidi, Marie-Francine Moens, Aaron Levine, Seth Dworman, and Zachary Yocum. 2015. SemEval-2015 Task 8: SpaceEval. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 884–894.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Stefanie A. Tellex, Thomas Fleming Kollar, Steven R. Dickerson, Matthew R. Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*.

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J. Mankowitz, and Shie Mannor. 2016. A Deep Hierarchical Approach to Lifelong Learning in Minecraft. *arXiv:1604.07255 [cs]*.