Learning Spatial Knowledge for Text to 3D Scene Generation

Angel X. Chang, Manolis Savva and Christopher D. Manning
Stanford University
{angelx,msavva,manning}@cs.stanford.edu

Abstract

We address the grounding of natural language to concrete spatial constraints, and inference of implicit pragmatics in 3D environments. We apply our approach to the task of text-to-3D scene generation. We present a representation for common sense spatial knowledge and an approach to extract it from 3D scene data. In text-to-3D scene generation, a user provides as input natural language text from which we extract explicit constraints on the objects that should appear in the scene. The main innovation of this work is to show how to augment these explicit constraints with learned spatial knowledge to infer missing objects and likely layouts for the objects in the scene. We demonstrate that spatial knowledge is useful for interpreting natural language and show examples of learned knowledge and generated 3D scenes.

1 Introduction

To understand language, we need an understanding of the world around us. Language describes the world and provides symbols with which we represent meaning. Still, much knowledge about the world is so obvious that it is rarely explicitly stated. It is uncommon for people to state that chairs are usually on the floor and upright, and that you usually eat a cake from a plate on a table. Knowledge of such common facts provides the context within which people communicate with language. Therefore, to create practical systems that can interact with the world and communicate with people, we need to leverage such knowledge to interpret language in context.

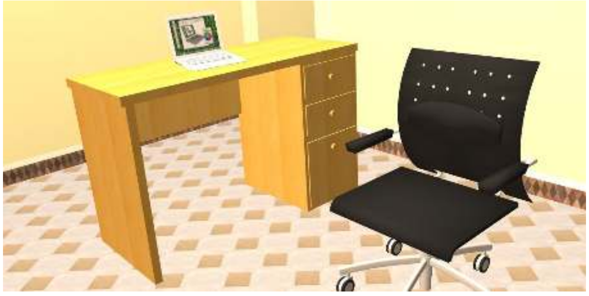


Figure 1: Generated scene for "There is a room with a chair and a computer." Note that the system infers the presence of a desk and that the computer should be supported by the desk.

Spatial knowledge is an important aspect of the world and is often not expressed explicitly in natural language. This is one of the biggest challenges in grounding language and enabling natural communication between people and intelligent systems. For instance, if we want a robot that can follow commands such as "bring me a piece of cake", it needs to be imparted with an understanding of likely locations for the cake in the kitchen and that the cake should be placed on a plate.

The pioneering WordsEye system (Coyne and Sproat, 2001) addressed the text-to-3D task and is an inspiration for our work. However, there are many remaining gaps in this broad area. Among them, there is a need for research into learning spatial knowledge representations from data, and for connecting them to language. Representing unstated facts is a challenging problem unaddressed by prior work and the focus of our contribution. This problem is a counterpart to the image description problem (Kulkarni et al., 2011; Mitchell et al., 2012; Elliott and Keller, 2013), which has so far remained largely unexplored by the community.

We present a representation for this form of spatial knowledge that we learn from 3D scene data and connect to natural language. We will show how this representation is useful for grounding language and for inferring unstated facts, i.e., the pragmatics of language describing physical environments. We demonstrate the use of this representation in the task of text-to-3D scene genera-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2028–2038,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

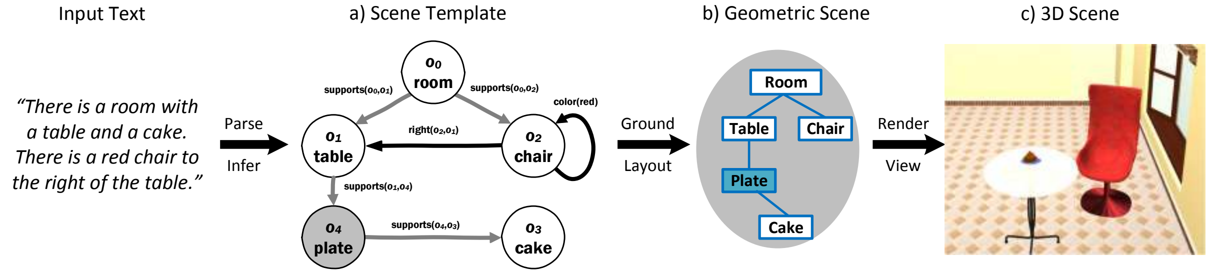


Figure 2: Overview of our spatial knowledge representation for text-to-3D scene generation. We parse input text into a scene template and infer implicit spatial constraints from learned priors. We then ground the template to a geometric scene, choose 3D models to instantiate and arrange them into a final 3D scene.

tion, where the input is natural language and the desired output is a 3D scene.

We focus on the text-to-3D task to demonstrate that extracting spatial knowledge is possible and beneficial in a challenging scenario: one requiring the grounding of natural language and inference of rarely mentioned implicit pragmatics based on spatial facts. Figure 1 illustrates some of the inference challenges in generating 3D scenes from natural language: the desk was not explicitly mentioned in the input, but we need to infer that the computer is likely to be supported by a desk rather than directly placed on the floor. Without this inference, the user would need to be much more verbose with text such as "There is a room with a chair, a computer, and a desk. The computer is on the desk, and the desk is on the floor. The chair is on the floor."

Contributions We present a spatial knowledge representation that can be learned from 3D scenes and captures the statistics of what objects occur in different scene types, and their spatial positions relative to each other. In addition, we model spatial relations (left, on top of, etc.) and learn a mapping between language and the geometric constraints that spatial terms imply. We show that using our learned spatial knowledge representation, we can infer implicit constraints, and generate plausible scenes from concise natural text input.

2 Task Definition and Overview

We define text-to-scene generation as the task of taking text that describes a scene as input, and generating a plausible 3D scene described by that text as output. More concretely, based on the input text, we select objects from a dataset of 3D models and arrange them to generate output scenes.

The main challenge we address is in transforming a scene template into a physically realizable 3D scene. For this to be possible, the system must be able to automatically specify the objects present and their position and orientation with respect to each other as constraints in 3D space. To do so, we need to have a representation of scenes (§3). We need good priors over the arrangements of objects in scenes (§4) and we need to be able to ground textual relations into spatial constraints (§5). We break down our task as follows (see Figure 2):

Template Parsing (§6.1): Parse the textual description of a scene into a set of constraints on the objects present and spatial relations between them.

Inference (§6.2): Expand this set of constraints by accounting for implicit constraints not specified in the text using learned spatial priors.

Grounding (§6.3): Given the constraints and priors on the spatial relations of objects, transform the scene template into a geometric 3D scene with a set of objects to be instantiated.

Scene Layout (§6.4): Arrange the objects and optimize their placement based on priors on the relative positions of objects and explicitly provided spatial constraints.

3 Scene Representation

To capture the objects present and their arrangement, we represent scenes as graphs where nodes are objects in the scene, and edges are semantic relationships between the objects.

We represent the semantics of a scene using a *scene template* and the geometric properties using a *geometric scene*. One critical property which is captured by our scene graph representation is that of a static support hierarchy, i.e., the order in which bigger objects physically support smaller ones: the floor supports tables, which support plates, which can support cakes. Static support and other constraints on relationships between objects are represented as edges in the scene graph.

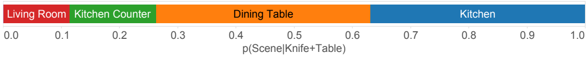


Figure 3: Probabilities of different scene types given the presence of “knife” and “table”.

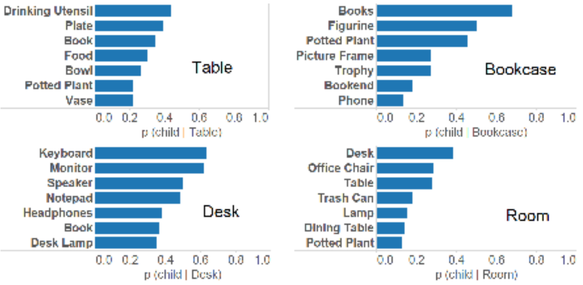


Figure 4: Probabilities of support for some most likely child object categories given four different parent object categories, from top left clockwise: dining table, bookcase, room, desk.

3.1 Scene Template

A scene template $\mathcal{T} = (\mathcal{O}, \mathcal{C}, C_s)$ consists of a set of object descriptions $\mathcal{O} = \{o_1, \ldots, o_n\}$ and constraints $\mathcal{C} = \{c_1, \ldots, c_k\}$ on the relationships between the objects. A scene template also has a scene type C_s.

Each object o_i, has properties associated with it such as category label, basic attributes such as color and material, and number of occurrences in the scene. For constraints, we focus on spatial relations between objects, expressed as predicates of the form *supported_by*(o_i, o_j) or *left*(o_i, o_j) where o_i and o_j are recognized objects.[1] Figure 2a shows an example scene template. From the scene template we instantiate concrete geometric 3D scenes. To infer implicit constraints on objects and spatial support we learn priors on object occurrences in 3D scenes (§4.1) and their support hierarchies (§4.2).

3.2 Geometric Scene

We refer to the concrete geometric representation of a scene as a “geometric scene”. It consists of a set of 3D model instances – one for each object – that capture the appearance of the object. A transformation matrix that represents the position, orientation, and scaling of the object in a scene is also necessary to exactly position the object. We generate a geometric scene from a scene template by selecting appropriate models from a 3D model database and determining transformations that optimize their layout to satisfy spatial constraints. To inform geometric arrangement we learn priors on the types of support surfaces (§4.2) and the relative positions of objects (§4.4).

4 Spatial Knowledge

Our model of spatial knowledge relies on the idea of abstract scene types describing the occurrence and arrangement of different categories of objects within scenes of that type. For example, kitchens typically contain kitchen counters on which plates and cups are likely to be found. The type of scene and category of objects condition the spatial relationships that can exist in a scene.

We learn spatial knowledge from 3D scene data, basing our approach on that of Fisher et al. (2012) and using their dataset of 133 small indoor scenes created with 1723 Trimble 3D Warehouse models (Fisher et al., 2012).

4.1 Object Occurrence Priors

We learn priors for object occurrence in different scene types (such as kitchens, offices, bedrooms).

$$P_{occ}(C_o|C_s) = \frac{count(C_o \text{ in } C_s)}{count(C_s)}$$

This allows us to evaluate the probability of different scene types given lists of object occurring in them (see Figure 3). For example given input of the form “there is a knife on the table” then we are likely to generate a scene with a dining table and other related objects.

4.2 Support Hierarchy Priors

We observe the static support relations of objects in existing scenes to establish a prior over what objects go on top of what other objects. As an example, by observing plates and forks on tables most of the time, we establish that tables are more likely to support plates and forks than chairs. We estimate the probability of a parent category C_p supporting a given child category C_c as a simple conditional probability based on normalized observation counts.[2]

$$P_{support}(C_p|C_c) = \frac{count(C_c \text{ on } C_p)}{count(C_c)}$$

We show a few of the priors we learn in Figure 4 as likelihoods of categories of child objects being statically supported by a parent category object.

[1] Our representation can also support other relationships such as $larger(o_i, o_j)$.

[2] The support hierarchy is explicitly modeled in the scene dataset we use.

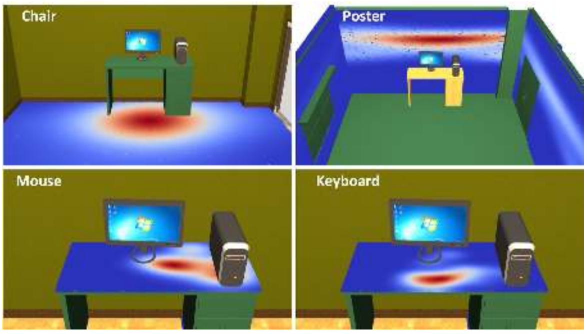


Figure 5: Predicted positions using learned relative position priors for *chair* given *desk* (top left), *poster-room* (top right), *mouse-desk* (bottom left), *keyboard-desk* (bottom right).

4.3 Support Surface Priors

To identify which surfaces on parent objects support child objects, we first segment parent models into planar surfaces using a simple region-growing algorithm based on (Kalvin and Taylor, 1996). We characterize support surfaces by the direction of their normal vector, limited to the six canonical directions: *up, down, left, right, front, back.* We learn a probability of supporting surface normal direction S_n given child object category C_c. For example, posters are typically found on walls so their support normal vectors are in the horizontal directions. Any unobserved child categories are assumed to have $P_{surf}(S_n = up|C_c) = 1$ since most things rest on a horizontal surface (e.g., floor).

$$P_{surf}(S_n|C_c) = \frac{count(C_c \text{ on surface with } S_n)}{count(C_c)}$$

4.4 Relative Position Priors

We model the relative positions of objects based on their object categories and current scene type: i.e., the relative position of an object of category C_{obj} is with respect to another object of category C_{ref} and for a scene type C_s. We condition on the relationship R between the two objects, whether they are siblings ($R = Sibling$) or child-parent ($R = ChildParent$).

$$P_{relpos}(x, y, \theta|C_{obj}, C_{ref}, C_s, R)$$

When positioning objects, we restrict the search space to points on the selected support surface. The position x, y is the centroid of the target object projected onto the support surface in the semantic frame of the reference object. The θ is the angle between the front of the two objects. We represent these relative position and orientation priors by performing kernel density estimation on the observed samples. Figure 5 shows predicted positions of objects using the learned priors.

Relation	$P(relation)$
inside(A,B)	$\frac{Vol(A\cap B)}{Vol(A)}$
outside(A,B)	1 - $\frac{Vol(A\cap B)}{Vol(A)}$
left_of(A,B)	$\frac{Vol(A\cap \text{left\_of }(B))}{Vol(A)}$
right_of(A,B)	$\frac{Vol(A\cap \text{right\_of }(B))}{Vol(A)}$
near(A,B)	$\mathbb{1}(dist(A, B) < t_{near})$
faces(A,B)	cos$(front(A), c(B) - c(A))$

Table 1: Definitions of spatial relation using bounding boxes. Note: $dist(A, B)$ is normalized against the maximum extent of the bounding box of B. $front(A)$ is the direction of the front vector of A and $c(A)$ is the centroid of A.

Keyword	*Top Relations and Scores*
behind	$(back\_of, 0.46), (back\_side, 0.33)$
adjacent	$(front\_side, 0.27), (outside, 0.26)$
below	$(below, 0.59), (lower\_side, 0.38)$
front	$(front\_of, 0.41), (front\_side, 0.40)$
left	$(left\_side, 0.44), (left\_of, 0.43)$
above	$(above, 0.37), (near, 0.30)$
opposite	$(outside, 0.31), (next\_to, 0.30)$
on	$(supported\_by, 0.86), (on\_top\_of, 0.76)$
near	$(outside, 0.66), (near, 0.66)$
next	$(outside, 0.49), (near, 0.48)$
under	$(supports, 0.62), (below, 0.53)$
top	$(supported\_by, 0.65), (above, 0.61)$
inside	$(inside, 0.48), (supported\_by, 0.35)$
right	$(right\_of, 0.50), (lower\_side, 0.38)$
beside	$(outside, 0.45), (right\_of, 0.45)$

Table 2: Map of top keywords to spatial relations (appropriate mappings in **bold**).

5 Spatial Relations

We define a set of formal spatial relations that we map to natural language terms (§5.1). In addition, we collect annotations of spatial relation descriptions from people, learn a mapping of spatial keywords to our formal spatial relations, and train a classifier that given two objects can predict the likelihood of a spatial relation holding (§5.2).

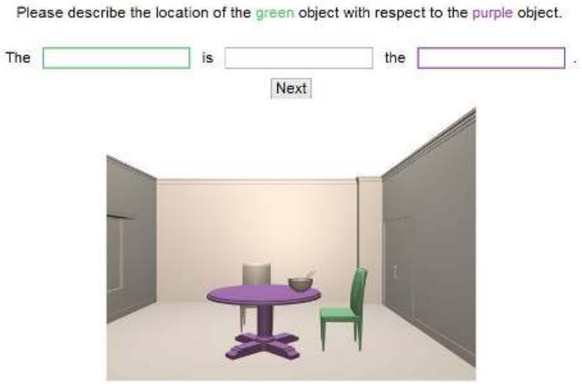
5.1 Predefined spatial relations

For spatial relations we use a set of predefined relations: *left_of, right_of, above, below, front, back, supported_by, supports, next_to, near, inside, outside, faces, left_side, right_side*.[3] These are measured using axis-aligned bounding boxes from the viewer's perspective; the involved bounding boxes are compared to determine volume overlap or closest distance (for proximity relations; see Table 1).

[3] We distinguish *left_of(A,B)* as A being left of the left edge of the bounding box of B vs *left_side(A,B)* as A being left of the centroid of B.

Feature	#	*Description*
$delta(A, B)$	3	Delta position (x, y, z) between the centroids of A and B
$dist(A, B)$	1	Normalized distance (wrt B) between the centroids of A and B
$overlap(A, f(B))$	6	Fraction of A inside left/right/front/back/top/bottom regions wrt B: $\frac{Vol(A \cap f(B))}{Vol(A)}$
$overlap(A, B)$	2	$\frac{Vol(A \cap B)}{Vol(A)}$ and $\frac{Vol(A \cap B)}{Vol(B)}$
$support(A, B)$	2	$supported\_by(A, B)$ and $supports(A, B)$

Table 3: Features for trained spatial relations predictor.



Please describe the location of the green object with respect to the purple object.

The □ is □ the □.

Next

Figure 6: Our data collection task.

Since these spatial relations are resolved with respect to the view of the scene, they correspond to view-centric definitions of spatial concepts.

5.2 Learning Spatial Relations

We collect a set of text descriptions of spatial relationships between two objects in 3D scenes by running an experiment on Amazon Mechanical Turk. We present a set of screenshots of scenes in our dataset that highlight particular pairs of objects and we ask people to fill in a spatial relationship of the form "The __ is __ the __" (see Fig 6). We collected a total of 609 annotations over 131 object pairs in 17 scenes. We use this data to learn priors on view-centric spatial relation terms and their concrete geometric interpretation.

For each response, we select one keyword from the text based on length. We learn a mapping of the top 15 keywords to our predefined set of spatial relations. We use our predefined relations on annotated spatial pairs of objects to create a binary indicator vector that is set to 1 if the spatial relation holds, or zero otherwise. We then create a similar vector for whether the keyword appeared in the annotation for that spatial pair, and then compute the cosine similarity of the two vectors to obtain a score for mapping keywords to spatial relations. Table 2 shows the obtained mapping. Using just the top mapping, we are able to map 10 of the 15 keywords to an appropriate spatial relation. The 5 keywords that are not well mapped are proximity relations that are not well captured by our predefined spatial relations.

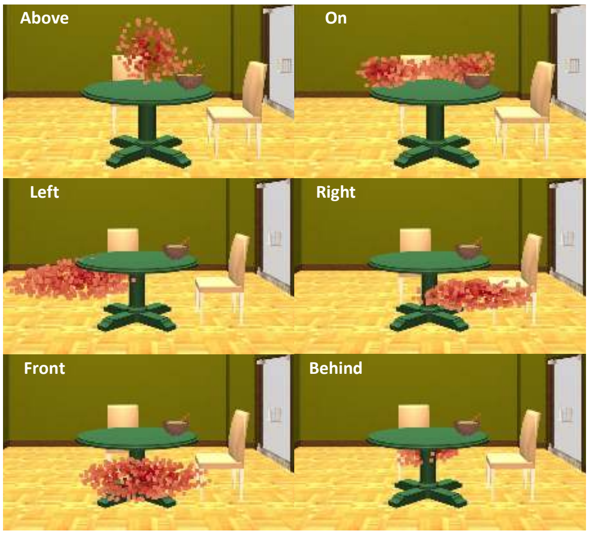


Figure 7: High probability regions where the center of another object would occur for some spatial relations with respect to a table: *above* (top left), *on* (top right), *left* (mid left), *right* (mid right), *in front* (bottom left), *behind* (bottom right).

Using the 15 keywords as our spatial relations, we train a log linear binary classifier for each keyword over features of the objects involved in that spatial relation (see Table 3). We then use this model to predict the likelihood of that spatial relation in new scenes.

Figure 7 shows examples of predicted likelihoods for different spatial relations with respect to an anchor object in a scene. Note that the learned spatial relations are much stricter than our predefined relations. For instance, "above" is only used to referred to the area directly above the table, not to the region above and to the left or above and in front (which our predefined classifier will all consider to be above). In our results, we show we have more accurate scenes using the trained spatial relations than the predefined ones.

Dependency Pattern	*Example Text*
`{tag:VBN}=verb >nsubjpass {}=nsubj >prep ({}=prep >pobj {}=pobj)`	The chair$_{\texttt{[nsubj]}}$ is made$_{\texttt{[verb]}}$ of$_{\texttt{[prep]}}$ wood$_{\texttt{[pobj]}}$.
attribute`(verb,pobj)(nsubj,pobj)`	`material(chair,wood)`
`{}=dobj >cop {} >nsubj {}=nsubj`	The chair$_{\texttt{[nsubj]}}$ is red$_{\texttt{[dobj]}}$.
attribute`(dobj)(nsubj,dobj)`	`color(chair,red)`
`{}=dobj >cop {} >nsubj {}=nsubj >prep ({}=prep >pobj {}=pobj)`	The table$_{\texttt{[nsubj]}}$ is next$_{\texttt{[dobj]}}$ to$_{\texttt{[prep]}}$ the chair$_{\texttt{[pobj]}}$.
spatial`(dobj)(nsubj, pobj)`	`next_to(table,chair)`
`{}=nsubj >advmod ({}=advmod >prep ({}=prep >pobj {}=pobj))`	There is a table$_{\texttt{[nsubj]}}$ next$_{\texttt{[advmod]}}$ to$_{\texttt{[prep]}}$ a chair$_{\texttt{[pobj]}}$.
spatial`(advmod)(nsubj, pobj)`	`next_to(table,chair)`

Table 4: Example dependency patterns for extracting attributes and spatial relations.

6 Text to Scene generation

We generate 3D scenes from brief scene descriptions using our learned priors.

6.1 Scene Template Parsing

During scene template parsing we identify the scene type, the objects present in the scene, their attributes, and the relations between them. The input text is first processed using the Stanford CoreNLP pipeline (Manning et al., 2014). The scene type is determined by matching the words in the utterance against a list of known scene types from the scene dataset.

To identify objects, we look for noun phrases and use the head word as the category, filtering with WordNet (Miller, 1995) to determine which objects are visualizable (under the physical object synset, excluding locations). We use the Stanford coreference system to determine when the same object is being referred to.

To identify properties of the objects, we extract other adjectives and nouns in the noun phrase. We also match dependency patterns such as “X is made of Y” to extract additional attributes. Based on the object category and attributes, and other words in the noun phrase mentioning the object, we identify a set of associated keywords to be used later for querying the 3D model database.

Dependency patterns are also used to extract spatial relations between objects (see Table 4 for some example patterns). We use Semgrex patterns to match the input text to dependencies (Chambers et al., 2007). The attribute types are determined from a dictionary using the text expressing the attribute (e.g., *attribute*(red)=color, *attribute*(round)=shape). Likewise, spatial relations are looked up using the learned map of keywords to spatial relations.

As an example, given the input “There is a room with a desk and a red chair. The chair is to the left of the desk.” we extract the following objects and spatial relations:

Objects	category	attributes	keywords
o_0	room		room
o_1	desk		desk
o_2	chair	*color*:red	chair, red

Relations: left(o_2, o_1)

6.2 Inferring Implicits

From the parsed scene template, we infer the presence of additional objects and support constraints.

We can optionally infer the presence of additional objects from object occurrences based on the scene type. If the scene type is unknown, we use the presence of known object categories to predict the most likely scene type by using Bayes' rule on our object occurrence priors P_{occ} to get $P(C_s|\{C_o\}) \propto P_{occ}(\{C_o\}|C_s)P(C_s)$. Once we have a scene type C_s, we sample P_{occ} to find objects that are likely to occur in the scene. We restrict sampling to the top $n = 4$ object categories.

We can also use the support hierarchy priors $P_{support}$ to infer implicit objects. For instance, for each object o_i we find the most likely supporting object category and add it to our scene if not already present.

After inferring implicit objects, we infer the support constraints. Using the learned text to predefined relation mapping from §5.2, we can map the keywords “on” and “top” to the *supported_by* relation. We infer the rest of the support hierarchy by selecting for each object o_i the parent object o_j that maximizes $P_{support}(C_{o_j}|C_{o_i})$.

6.3 Grounding Objects

Once we determine from the input text what objects exist and their spatial relations, we select 3D models matching the objects and their associated properties. Each object in the scene template is grounded by querying a 3D models database with

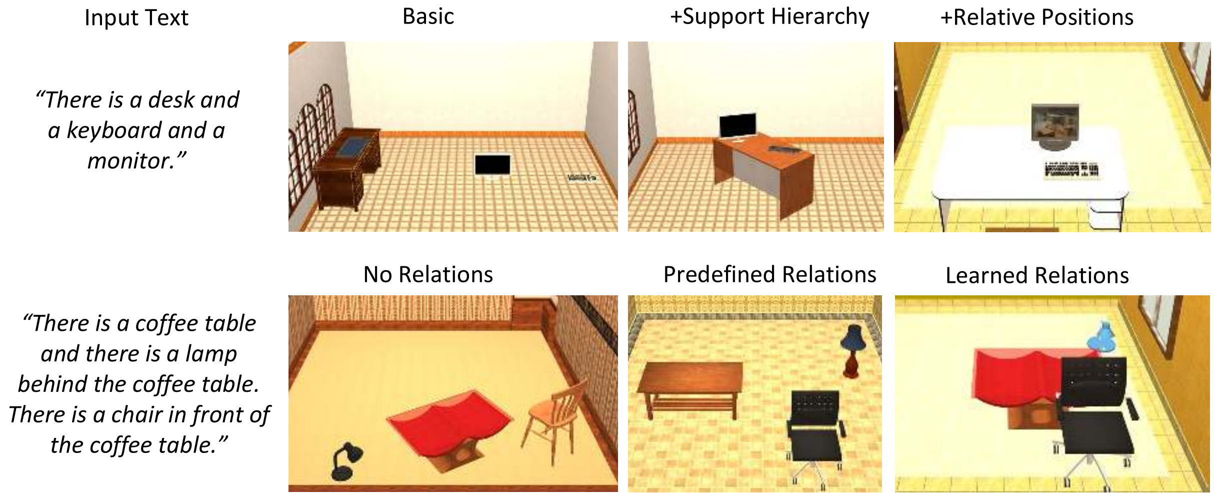

Input Text
Basic
+Support Hierarchy
+Relative Positions
"There is a desk and a keyboard and a monitor."
No Relations
Predefined Relations
Learned Relations
"There is a coffee table and there is a lamp behind the coffee table. There is a chair in front of the coffee table."

Figure 8: **Top** Generated scenes for randomly placing objects on the floor (*Basic*), with inferred *Support Hierarchy*, and with priors on *Relative Positions*. **Bottom** Generated scenes with no understanding of spatial relations (*No Relations*), scoring using *Predefined Relations* and *Learned Relations*.

the appropriate category and keywords.

We use a 3D model dataset collected from Google 3D Warehouse by prior work in scene synthesis and containing about 12490 mostly indoor objects (Fisher et al., 2012). These models have text associated with them in the form of names and tags. In addition, we semi-automatically annotated models with object category labels (roughly 270 classes). We used model tags to set these labels, and verified and augmented them manually.

In addition, we automatically rescale models so that they have physically plausible sizes and orient them so that they have a consistent up and front direction (Savva et al., 2014). We then indexed all models in a database that we query at run-time for retrieval based on category and tag labels.

6.4 Scene Layout

Once we have instantiated the objects in the scene by selecting models, we aim to optimize an overall layout score $\mathcal{L} = \lambda_{obj}\mathcal{L}_{obj} + \lambda_{rel}\mathcal{L}_{rel}$ that is a weighted sum of object arrangement $\mathcal{L}_{obj}$ score and constraint satisfaction $\mathcal{L}_{rel}$ score:

$$\mathcal{L}_{obj} = \sum_{o_i} P_{surf}(S_n|C_{o_i}) \sum_{o_j \in F(o_i)} P_{relpos}(\cdot)$$

$$\mathcal{L}_{rel} = \sum_{c_i} P_{rel}(c_i)$$

where $F(o_i)$ are the sibling objects and parent object of o_i. We use $\lambda_{obj} = 0.25$ and $\lambda_{rel} = 0.75$ for the results we present.

We use a simple hill climbing strategy to find a reasonable layout. We first initialize the positions of objects within the scene by traversing the support hierarchy in depth-first order, positioning the children from largest to first and recursing. Child nodes are positioned by first selecting a supporting surface on a candidate parent object through sampling of P_{surf}. After selecting a surface, we sample a position on the surface based on P_{relpos}. Finally, we check whether collisions exist with other objects, rejecting layouts where collisions occur. We iterate by randomly jittering and repositioning objects. If there are any spatial constraints that are not satisfied, we also remove and randomly reposition the objects violating the constraints, and iterate to improve the layout. The resulting scene is rendered and presented to the user.

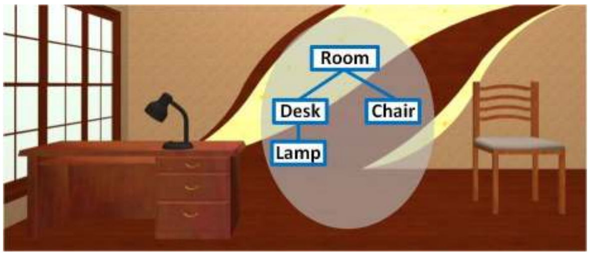


Figure 9: Generated scene for "There is a room with a desk and a lamp. There is a chair to the right of the desk." The inferred scene hierarchy is overlayed in the center.

7 Results and Discussion

We show examples of generated scenes, and compare against naive baselines to demonstrate learned priors are essential for scene generation. We

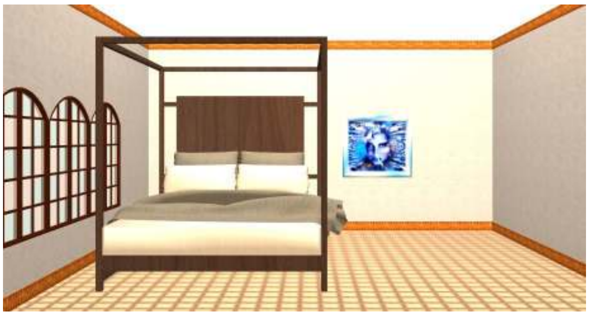


Figure 10: Generated scene for "There is a room with a poster bed and a poster."

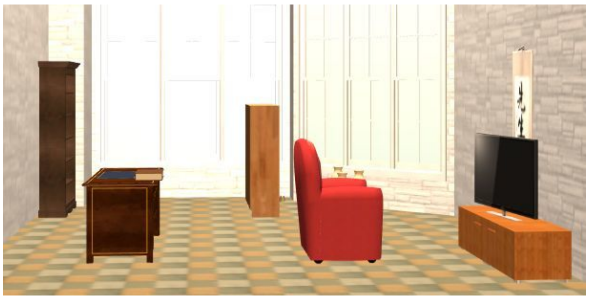


Figure 11: Generated scene for "living room".

also discuss interesting aspects of using spatial knowledge in view-based object referent resolution (§7.2) and in disambiguating geometric interpretations of "on" (§7.3).

Model Comparison Figure 8 shows a comparison of scenes generated by our model versus several simpler baselines. The top row shows the impact of modeling the support hierarchy and the relative positions in the layout of the scene. The bottom row shows that the learned spatial relations can give a more accurate layout than the naive predefined spatial relations, since it captures pragmatic implicatures of language, e.g., left is only used for directly left and not top left or bottom left (Vogel et al., 2013).

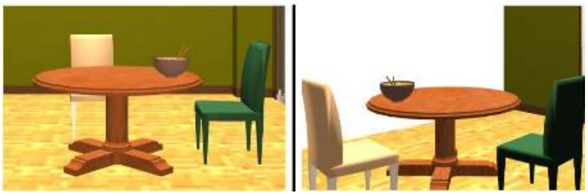


Figure 12: **Left**: chair is selected using "the chair to the right of the table" or "the object to the right of the table". Chair is not selected for "the cup to the right of the table". **Right**: Different view results in different chair being selected for the input "the chair to the right of the table".

7.1 Generated Scenes

Support Hierarchy Figure 9 shows a generated scene along with the input text and support hierarchy. Even though the spatial relation between lamp and desk was not mentioned, we infer that the lamp is supported by the top surface of the desk.

Disambiguation Figure 10 shows a generated scene for the input "There is a room with a poster bed and a poster". Note that the system differentiates between a "poster" and a "poster bed" – it correctly selects and places the bed on the floor, while the poster is placed on the wall.

Inferring objects for a scene type Figure 11 shows an example of inferring all the objects present in a scene from the input "living room". Some of the placements are good, while others can clearly be improved.

7.2 View-centric object referent resolution

After a scene is generated, the user can refer to objects with their categories and with spatial relations between them. Objects are disambiguated by both category and view-centric spatial relations. We use the WordNet hierarchy to resolve hyponym or hypernym referents to objects in the scene. In Figure 12 (left), the user can select a chair to the right of the table using the phrase "chair to the right of the table" or "object to the right of the table". The user can then change their viewpoint by rotating and moving around. Since spatial relations are resolved with respect to the current viewpoint, a different chair is selected for the same phrase from a different viewpoint in the right screenshot.

7.3 Disambiguating "on"

As shown in §5.2, the English preposition "on", when used as a spatial relation, corresponds strongly to the *supported_by* relation. In our trained model, the *supported_by* feature also has a high positive weight for "on".

Our model for supporting surfaces and hierarchy allows interpreting the placement of "A on B" based on the categories of A and B. Figure 13 demonstrates four different interpretations for "on". Given the input "There is a cup on the table" the system correctly places the cup on the top surface of the table. In contrast, given "There is a cup on the bookshelf", the cup is placed on a supporting surface of the bookshelf, but not necessarily the top one which would be fairly high.

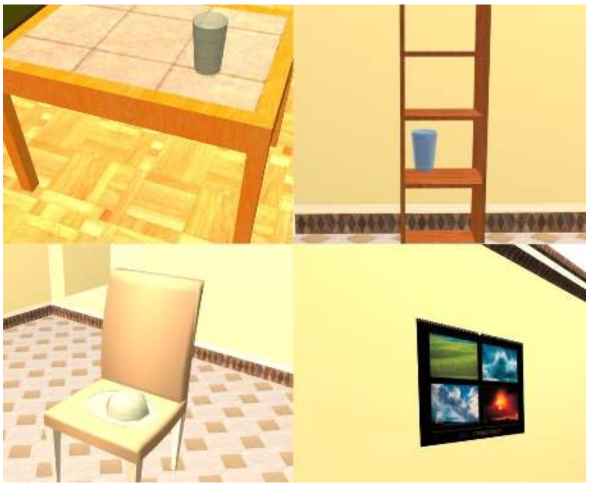


Figure 13: From top left clockwise: "There is a cup on the table", "There is a cup on the bookshelf", "There is a poster on the wall", "There is a hat on the chair". Note the different geometric interpretations of "on".

Given the input "There is a poster on the wall", a poster is pasted on the wall, while with the input "There is a hat on the chair" the hat is placed on the seat of the chair.

7.4 Limitations

While the system shows promise, there are still many challenges in text-to-scene generation. For one, we did not address the difficulties of resolving objects. A failure case of our system stems from using a fixed set of categories to identify visualizable objects. For example, the sense of "top" referring to a spinning top, and other uncommon object types, are not handled by our system as concrete objects. Furthermore, complex phrases including object parts such as "there's a coat on the seat of the chair" are not handled. Figure 14 shows some example cases where the context is important in selecting an appropriate object and the difficulties of interpreting noun phrases.

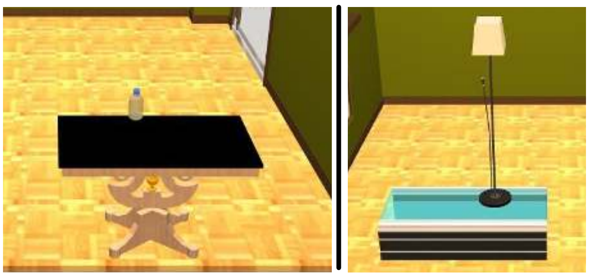


Figure 14: **Left**: A water bottle instead of wine bottle is selected for "There is a bottle of wine on the table in the kitchen". In addition, the selected table is inappropriate for a kitchen. **Right**: A floor lamp is incorrectly selected for the input "There is a lamp on the table".

In addition, we rely on a few dependency patterns for extracting spatial relations so robustness to variations in spatial language is lacking. We only handle binary spatial relations (e.g., "left", "behind") ignoring more complex relations such as "around the table" or "in the middle of the room". Though simple binary relations are some of the most fundamental spatial expressions and a good first step, handling more complex expressions will do much to improve the system.

Another issue is that the interpretation of sentences such as "the desk is covered with paper", which entails many pieces of paper placed on the desk, is hard to resolve. With a more data-driven approach we can hope to link such expressions to concrete facts.

Finally, we use a traditional pipeline approach for text processing, so errors in initial stages can propagate downstream. Failures in dependency parsing, part of speech tagging, or coreference resolution can result in incorrect interpretations of the input language. For example, in the sentence "there is a desk with a chair in front of it", "it" is not identified as coreferent with "desk" so we fail to extract the spatial relation $\texttt{front\_of}(\texttt{chair}, \texttt{desk})$.

8 Related Work

There is related prior work in the topics of modeling spatial relations, generating 3D scenes from text, and automatically laying out 3D scenes.

8.1 Spatial knowledge and relations

Prior work that required modeling spatial knowledge has defined representations specific to the task addressed. Typically, such knowledge is manually provided or crowdsourced – not learned from data. For instance, WordsEye (Coyne et al., 2010) uses a set of manually specified relations. The NLP community has explored grounding text to physical attributes and relations (Matuszek et al., 2012; Krishnamurthy and Kollar, 2013), generating text for referring to objects (FitzGerald et al., 2013) and connecting language to spatial relationships (Vogel and Jurafsky, 2010; Golland et al., 2010; Artzi and Zettlemoyer, 2013). Most of this work focuses on learning a mapping from text to formal representations, and does not model implicit spatial knowledge. Many priors on real world spatial facts are typically unstated in text and remain largely unaddressed.

8.2 Text to Scene Systems

Early work on the SHRDLU system (Winograd, 1972) gives a good formalization of the linguistic manipulation of objects in 3D scenes. By restricting the discourse domain to a micro-world with simple geometric shapes, the SHRDLU system demonstrated parsing of natural language input for manipulating scenes. However, generalization to more complex objects and spatial relations is still very hard to attain.

More recently, a pioneering text-to-3D scene generation prototype system has been presented by WordsEye (Coyne and Sproat, 2001). The authors demonstrated the promise of text to scene generation systems but also pointed out some fundamental issues which restrict the success of their system: much spatial knowledge is required which is hard to obtain. As a result, users have to use unnatural language (e.g., “the stool is 1 feet to the south of the table”) to express their intent. Follow up work has attempted to collect spatial knowledge through crowd-sourcing (Coyne et al., 2012), but does not address the learning of spatial priors.

We address the challenge of handling natural language for scene generation, by learning spatial knowledge from 3D scene data, and using it to infer unstated implicit constraints. Our work is similar in spirit to recent work on generating 2D clipart for sentences using probabilistic models learned from data (Zitnick et al., 2013).

8.3 Automatic Scene Layout

Work on scene layout has focused on determining good furniture layouts by optimizing energy functions that capture the quality of a proposed layout. These energy functions are encoded from design guidelines (Merrell et al., 2011) or learned from scene data (Fisher et al., 2012). Knowledge of object co-occurrences and spatial relations is represented by simple models such as mixtures of Gaussians on pairwise object positions and orientations. We leverage ideas from this work, but they do not focus on linking spatial knowledge to language.

9 Conclusion and Future Work

We have demonstrated a representation of spatial knowledge that can be learned from 3D scene data and how it corresponds to natural language. We also showed that spatial inference and grounding is critical for achieving plausible results in the text-to-3D scene generation task. Spatial knowledge is critically useful not only in this task, but also in other domains which require an understanding of the pragmatics of physical environments.

We only presented a deterministic approach for mapping input text to the parsed scene template. An interesting avenue for future research is to automatically learn how to parse text describing scenes into formal representations by using more advanced semantic parsing methods.

We can also improve the representation used for spatial priors of objects in scenes. For instance, in this paper we represented support surfaces by their orientation. We can improve the representation by modeling whether a surface is an interior or exterior surface.

Another interesting line of future work would be to explore the influence of object identity in determining when people use ego-centric or object-centric spatial reference models, and to improve resolution of spatial terms that have different interpretations (e.g., “the chair to the left of John” vs “the chair to the left of the table”).

Finally, a promising line of research is to explore using spatial priors for resolving ambiguities during parsing. For example, the attachment of “next to” in “Put a lamp on the table next to the book” can be readily disambiguated with spatial priors such as the ones we presented.

Acknowledgments

We thank the anonymous reviewers for their thoughtful comments. We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*.

Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage, Eric Yeh, and Christopher D. Manning. 2007. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Bob Coyne and Richard Sproat. 2001. WordsEye: an automatic text-to-scene conversion system. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*.

Bob Coyne, Richard Sproat, and Julia Hirschberg. 2010. Spatial relations in text-to-scene conversion. In *Computational Models of Spatial Language Interpretation, Workshop at Spatial Cognition*.

Bob Coyne, Alexander Klapheke, Masoud Rouhizadeh, Richard Sproat, and Daniel Bauer. 2012. Annotation tools and knowledge representation for a text-to-scene system. *Proceedings of COLING 2012: Technical Papers*.

Desmond Elliott and Frank Keller. 2013. Image description using visual dependency representations. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. 2012. Example-based synthesis of 3D object arrangements. *ACM Transactions on Graphics (TOG)*.

Nicholas FitzGerald, Yoav Artzi, and Luke Zettlemoyer. 2013. Learning distributions over logical forms for referring expression generation. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Alan D. Kalvin and Russell H. Taylor. 1996. Superfaces: Polygonal mesh simplification with bounded error. *Computer Graphics and Applications, IEEE*.

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*.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. 2011. Baby talk: Understanding and generating simple image descriptions. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Cynthia Matuszek, Nicholas Fitzgerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *International Conference on Machine Learning (ICML)*.

Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. 2011. Interactive furniture layout using interior design guidelines. In *ACM Transactions on Graphics (TOG)*.

George A. Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*.

Margaret Mitchell, Xufeng Han, Jesse Dodge, Alyssa Mensch, Amit Goyal, Alex Berg, Kota Yamaguchi, Tamara Berg, Karl Stratos, and Hal Daumé III. 2012. Midge: Generating image descriptions from computer vision detections. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*.

Manolis Savva, Angel X. Chang, Gilbert Bernstein, Christopher D. Manning, and Pat Hanrahan. 2014. On being the right scale: Sizing large collections of 3D models. *Stanford University Technical Report CSTR 2014-03*.

Adam Vogel and Dan Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of ACL*.

Adam Vogel, Christopher Potts, and Dan Jurafsky. 2013. Implicatures and nested beliefs in approximate Decentralized-POMDPs. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Terry Winograd. 1972. Understanding natural language. *Cognitive psychology*.

C. Lawrence Zitnick, Devi Parikh, and Lucy Vanderwende. 2013. Learning the visual interpretation of sentences. In *IEEE Intenational Conference on Computer Vision (ICCV)*.