Abstract

In this ongoing work, we investigate how to build out the scene interaction component of the Text2Scene system. In particular, we enhance the interactivity of the scene generation process by providing a natural language front-end, whereby a user can provide textual commands to iteratively alter a scene. Further, we provide descriptions of the preliminary work in using advanced semantic parsing techniques to improve the process of extracting, interpreting, and executing the semantic meaning of user textual commands.

1 Introduction

Designing 3D scenes is a challenging task with a number of common use cases including furniture layout in catalogues, video game design, and visual storyboarding. Current 3D design tools provide a great amount of control over the construction and positioning of geometry within 3D scenes. However, these tools do not allow for intuitively assembling a scene from existing objects which is the most important capability for non-professional users. How might we expand the demographic of users with the ability to generate 3D scenes by providing a simple and robust natural language interface? We envision such an interface whereby a user could simply provide the textual command Create a kitchen with marble counter tops and a wooden chair and an appropriate 3D scene satisfying the constraints present in the command. Further, our interface would allow the user to continuously interact with the system to iteratively improve the scene until it adequately matches the user’s envisioned specifications.

Text to 3D scene systems face a number of difficult technical challenges. Firstly, natural language tends to be terse and incomplete. Hence, a user’s description of a desired scene, in addition to the explicit constraints mentioned in the text, may contain a number of implicit constraints that are not mentioned. Therefore, it falls on the system to adequately codify and interpret this common-sense knowledge and ensure that these constraints are met.

Secondly, people reason about the world at a much higher level than typical representations of 3D scenes (table against wall vs. 3D transformation matrix). The semantics of distinct objects and how their approximate arrangement allow human interaction are much more important than the precise and abstract geometrical properties of the objects. A functional text to 3D scene generation system must be able to seamlessly make the transition from high-level user-given semantic forms to the low-level machine-interpretable representations that are needed by computers.

This project builds on the Text2Scene system developed by Chang et. al. [2, 3, 4]. In particular, it aims to augment the interactivity of the existing system by providing support for iterative refinement through the use of scene interaction commands. We provide sensible semantic representations interpretable by Text2Scene for a number of common classes of textual commands. We also provide a description of preliminary work on using advanced semantic parsing techniques to extract relevant information from user input.

2 Background

Current research in human-computer interaction usually focuses on developing low-level interfaces for creating 3D scenes, whereby users have more ease in manipulating basic primitives such as curves [11]. However, there is relatively little work on direct 3D scene manipulation at a more broad semantic level.

Early work in textual interfaces for 3D scene manipulation has addressed simplified scenarios with micro-worlds consisting of simple geometric shapes. The SHRDLU [14] and PUT [5] systems were pioneers in parsing natural language instructions, but generalization of their approach to more realistic scenarios similar to the real world is challenging. More recent work on
the WordsEye system [6, 7] and other similar approaches [13] has demonstrated more general text-driven 3D scene generation that can handle complex scenes. The authors compellingly show the benefit of text to scene generation but note that their systems are restricted due to a lack of implicit spatial knowledge.

Recent work in correlating natural language with real-world knowledge by Matuszek et. al [10] has investigated the development of joint learning of language and perception models for grounded attribute induction. While their work is similar to ours in that they are able to select objects in scenes using descriptive attributes of the desired objects, their language model is based on probabilistic categorial grammars whereas our model attempts to implicitly learn logical forms from given question-answer pairs. Related work by Fitzgerald et. al. [8] seeks to tackle the problem of referring expression generation by learning appropriate logical forms from images of objects selected with select attributes. Our work is different in that we are seeking to solve the reverse problem: given a natural language utterance, identify objects in a scene with certain attributes.

3 Scene Interactions

To allow the user of our system to interact with a generated scene, we define a set of high-level semantic operations for scene manipulation. The provided text is treated as a set of constraints that we want the modified scene to satisfy, while trying to keep the revised scene as similar to the original scene as possible. More formally, given the original scene $S$ and a scene operation $O$, we want to find the scene $S'$ which is most similar to $S$, while still satisfying the constraints imposed by $O$.

To track elements of the scene, we maintain a scene state $Z = (S, Sel, Cam)$ that consists of the scene $S$, the set of selected objects $Sel$, and the camera position $Cam$. Each operation can be defined as a function $O : Z \rightarrow Z'$. We support several basic operations: $Select$ changes the set of selected objects, $Insert$ adds objects into the scene, $Delete$ removes objects from the scene, $Replace$ replaces objects in the scene with new objects, operations such as $Move$ and $Scale$ modify constraints on existing objects without changing the set of objects in the scene. $LookAt$ repositions the camera to focus on the selected objects. Figure 1 summarizes the operations supported by our system. These basic operations demonstrate some simple scene manipulations through natural language.

To interpret a textual scene interaction command, the system first parses the input text $u$ into a sequence of scene operations $(O_1, \ldots, O_k)$, identifying the resulting set of constraints that should hold. For each parsed scene operation $O_i$, the system then executes the scene operation by resolving the set of objects on which the operation should be performed and then modifying the scene state accordingly.

The text input currently is parsed by going through a rule-based natural language processing pipeline. Trigger verbs are deterministically mapped to possible actions, as shown in Table 1. Some verbs map to multiple actions, as in the case of “place” and “put” referring to either $Move$ and $Insert$. In the next section, we describe our preliminary work in using statistical semantic parsing techniques to parse the user utterance.

For a given operation, the objects in a scene are modified in the following manner:

- **Select**: Select the desired object and highlight it.
- **Insert**: Select a new object that satisfies the constraints from the model database and place it in the scene.
- **Replace**: Select a new object that satisfies the constraints from the model database and replace the objects to be replaced with the new object.
- **Remove**: Delete the old objects from the scene.

**Figure 1: Example Parses**

<table>
<thead>
<tr>
<th>Verb</th>
<th>Operation</th>
<th>Example text</th>
<th>Example parse</th>
</tr>
</thead>
<tbody>
<tr>
<td>select</td>
<td>Select</td>
<td>select the chair on the right of the table</td>
<td>$Select({lamp}, {right(lamp,table)})$</td>
</tr>
<tr>
<td>look, look at</td>
<td>LookAt</td>
<td>look at the lamp</td>
<td>$LookAt({lamp})$</td>
</tr>
<tr>
<td>add, insert, place, put</td>
<td>Insert</td>
<td>add a lamp to the table</td>
<td>$Insert({lamp}, {on(lamp,table)})$</td>
</tr>
<tr>
<td>delete, remove</td>
<td>Remove</td>
<td>remove the lamp</td>
<td>$Remove({lamp})$</td>
</tr>
<tr>
<td>replace</td>
<td>Replace</td>
<td>replace the lamp with a vase</td>
<td>$Replace({lamp}, {vase})$</td>
</tr>
<tr>
<td>move, place, put</td>
<td>Move</td>
<td>move the chair to the left</td>
<td>$Move({chair}, {left(chair)})$</td>
</tr>
<tr>
<td>enlarge, shrink</td>
<td>Scale</td>
<td>enlarge the bowl</td>
<td>$Scale({bowl})(1.5)$</td>
</tr>
</tbody>
</table>
• **Move**: Shift the object in the direction and to the degree specified by the text.

• **Scale**: Radially enlarge the object by a pre-defined factor.

1: There is a living room with a red couch and a TV.

2: Put a rug in the room.

3: Move the rug to the back.

4: Enlarge the rug.

Figure 2: Example Interactions

Once a set of objects is determined, the scene is altered so that all constraints are satisfied while minimizing changes to objects that are not selected. For example, in the case of the **Move** or **Scale** operations, the set of objects in the scene will remain the same but their relative sizes and positions will change to satisfy constraints. Figure 2 above shows an example of a series of user provided scene operations along with how they change the scene.

4 Semantic Parsing of Commands

The task of semantic parsing seeks to map natural language utterances to denotations via a series of intermediate logical forms. Traditional semantic parsers suffer from the extreme limitation that training the parsers requires having a dataset of hand-annotated logical forms as supervision. [15] [16] Typically this is very expensive training data to acquire. For the purposes of this work, we seek to use alternate semantic parsing methods introduced by Berant et. al. [1] in the form of their **Sempre** system. **Sempre** is a general-purpose semantic parsing framework that circumvents the need to provide explicitly annotated logical forms as supervision, but rather takes as input utterance-denotation training examples and infers the appropriate latent logical forms.

For our preliminary work, we have chosen to focus on the scene interaction **Select** operation. Upon successful application of the **Sempre** framework to the **Select** operation, we intend to use these techniques to provide support for the full set of scene operations defined above. In the next section, we describe the dataset we compiled for our learning problem.

4.1 Dataset

We obtained the training examples needed to build a semantic parser for our task using crowdsourcing methods. Amazon Mechanical Turkers were given two images. First Turkers would be presented a scene in full color, typically a close-up of some household setting such as a kitchen or a working desk. Afterwards, the Turkers would be presented the same scene with all objects grayed out except for one object highlighted in green. The Turkers were then told to provide the textual command that would select the item highlighted in the second image, making sure to provide relevant spatial descriptions of the object especially of its position relative to other objects in the scene. An example of a pair of images provided to Turkers as well as the corresponding description given by workers is provided in Figure 3.

Images provided to Turkers were taken from 111 distinct generated scene pairs, producing an initial dataset of 1174 image pairs with associated textual commands. This set was pruned further, removing all pairs where workers did not adequately understand and therefore complete the task. This left a cleaned dataset of 1084 image pairs. As our dataset is sampled from 111 distinct scene pairs, there are many repeated examples, where the same image pairs were provided to different Turkers. This is acceptable for our purposes as the utterances provided by different Turkers exhibit sufficient lexical and semantic diversity as to preserve the difficulty and richness of our learning problem. For each of the 111 distinct scene pairs, there are an average 9.7 different natural language utterances provided by different workers with a standard deviation of 2.3.

4.2 Learning a Grammar and Lexicon

As part of the **Sempre** pipeline, we are required to provide a grammar as well as a lexicon to assist the framework in generating latent logical forms from input utterances. The system will then learn to rank and appropriately score the logical forms that map
to the correct denotations through the use of user-provided linguistic and semantic features. It is up to the user’s discretion to decide how extensive of a grammar and lexicon will be automatically provided or how much the system will learn through statistical methods. Currently, our system utilizes a fairly rudimentary grammar that extracts noun entities in the utterance, using the CoreNLP pipeline [9], and then aims to map these entities to system-learned lexicon entries. Our grammar also identifies binary spatial relations among entities within an utterance, using trigger words such as under, faces, left, right, etc. These relations allow us to determine the relationship between certain entities in an utterance, which allows us to convert them to our internal semantic representation as exemplified in Figure 1.

Rather than manually providing a detailed lexicon with all the possible entities and object types, we seek to use a more statistical approach. In particular, our grammar currently has no notion of predefined lexical objects, such as table or chair. Instead, we provide the framework a set of semantic features to help it learn the association of table in the utterance “Select the brown table” to the 3D model of a ‘table’ present in the scene, assuming such a model exists. Our semantic features is based on distributed word vectors. Specifically, once our framework extracts all the nouns it uses the GLoVe [12] vector of the extracted entity and computes the cosine similarity to the GLoVe vectors of all the object categories in the scene, assuming an appropriate GLoVe entry exists. A feature is added for each such pairwise similarity measure. Currently it is not known how effective these features are in learning a lexicon, as we have not done a run of the system on a statistically significant portion of our dataset.

5 Future Work

At the moment, the work on applying Sempre to scene interactions for Text2Scene is still in its early stages. Therefore we have a number of future avenues of exploration. Firstly, we can augment the current features we are using to automatically learn a lexicon, making use of similarity as measured with tools such as WordNet or perhaps using the edit distance between lexical items. We can also augment our grammar to provide support for attribute extraction and usage, so that if a scene has multiple instances of an item such as a ‘chair’, we can utilize the attributes of the entity mentioned in a user utterance, such as Select the brown chair, behind the table. Finally, pending success with the Select operation, a natural next step is to apply the same techniques to the remaining classes of scene interactions mentioned in Section 3 above.

References


