A Language Visualisation System

Emre Unal and Deniz Yuret

Koc University,
Istanbul, Turkey
{emunal,dyuret}@ku.edu.tr
http://www.ku.edu.tr

Abstract. A novel language visualization system is presented here that is capable of understanding some concrete nouns, visualizable adjectives and spatial prepositions in full natural language sentences to generate 3D scenes. The system has a rule based question answering component and it can answer spatial inference questions about the scene created by these sentences. This work is the first step of our greater vision of building a sophisticated text-to-scene and scene-to-text conversion engine.

Keywords: language visualisation, language understanding, text-to-scene conversion, scene-to-text conversion, spatial inference

1 Introduction

Generating 3D scenes using natural language is a more practical and fun way to interact with a computer compared to classical methods of using specialized software and tools. These methods can be used in many practical applications like second language learning, story telling and games. We present a language visualization system that may serve these purposes. Beyond that we implemented a question answering engine that is capable of answering questions about the scene that is free of the limitations of just using shallow semantics to answer certain types of spatial inference problems. The system is rule based at its current stage, but it can be extended in the future to learn these connections from language to 3D scene and vice versa.

2 Prior Work

Natural language processing (NLP) algorithms are often used in conjunction with other techniques in domains like image processing, robotics and 3D graphics. Adding NLP components is critical as it provides a seamless human computer interaction. SHRDLU [2] is a system which uses natural language to interact with a robot living inside a virtual world with different blocks. Blocks can be moved and the state of the virtual world can be queried using natural language. SHRDLU is limited in the sense that all the actions are available and applied to the blocks that already exist in the virtual world. Put system [1] is an object placement system that uses natural language. It makes scene generation easier by using strict subset of English and direct manipulation. It generates scenes by taking into account different senses of the spatial relations. CarSim [3] is an automatic text-to-scene conversion system that simulates car accidents from written accident reports. Its NLP module is capable of understanding collision verbs and animates the vehicles in the scene accordingly. It generates scenes from written text to animations in a restricted domain successfully. CONFUCIUS [4] is a storytelling system that visualizes single sentences into 3D animations. It combines natural language and computer graphics techniques to generate complex high-level animations, sounds and speech. WordsEye [5] is a system for converting natural language into 3D scenes. Our scene construction work is inspired by it. However WordsEye focuses on creating visually more detailed scenes where we present a modest scene creation system and instead focus on question answering and scene to text conversion. As WordsEye, our system is rule based and depend on lexical resources that are mostly built manually. We believe that these rules can be learned with statistical machine learning algorithms however rule based approaches are good enough for establishing the scene construction framework. A recent work [6] combines computer vision with natural language processing in a grounded learning framework and uses learned word meanings to describe novel scenes. Although the processing of real images can lead to practical real life applications we believe that computer generated images are better to work with since you could generate many samples in a fraction of the time and they will be all noise free. These are huge advantages when put in a learning setup especially. Combining learning algorithms and scene construction with computer generated images is done before in DESCRIBER system [7] where randomly generated rectangles (random sizes, positions and colors) are matched with natural language descriptions of the selected rectangle. It has been shown that the machine can successfully describe newly seen rectangles. Our 3D scene construction work is more complex since we can have many different object types (528 models) corresponding to thousands of nouns so we decided to go with a rule based approach where we can setup an environment to establish a text to scene framework and on which new experiments can be run.

3 Components

This section describes several components our system is built upon and gives details about how each individual part contributes to it

3.1 Alice

Alice [8] is an open source programming environment to create animations using 3D models with an easy-to-use drag and drop interface. Alice is used for educational purposes mostly to teach problem solving, computational thinking and computer programming. For our language visualization system, we use model modification and rendering capabilities of Alice so that we can focus more on implementing language related features rather than re-implementing 3D scene manipulation functionalities. Alice also has an extensive gallery of 3D models consisting of many categories like people, animals, vehicles, furniture etc. In our system, we use 528 models from Alice gallery.

3.2 WordNet

WordNet [9] is a large lexical database of English that consists of nouns, verbs, adjectives and adverbs that are grouped into categories called synonym sets (synsets). Each synset represent a distinct concept (word sense) and these synsets are linked to others by semantic relations. These semantic relations are used heavily in our system, and they will be discussed further in the next section.

3.3 CoreNLP

CoreNLP [10] suite from Stanford provides many useful natural language analysis tools like parser, named entity recognizer, part of speech tagger and a coreference resolution system for English. These tools make a solid foundation for our language visualization system as they are vital for understanding written text. These tools are used in conjunction with our system to create accurate scenes from a written text input and then answering questions about the scene.

4 Implementation

Our language visualisation system is presented with a text box and a 3D virtual world to the user. Light, camera and ground are already added to the scene and the system waits for natural language text input from the user. After this stage, the scene can be constructed and manipulated directly by the natural language input. The next section discusses language components in our system before moving forward to scene construction and question answering as it forms a solid understanding for the inner workings of our system.

4.1 Language Components

Our system currently covers three main word types which are nouns, adjectives and prepositions. Dynamic verbs are intentionally left out at this stage as we limited ourselves to only static scenes. Even though they are covered by other systems using posing [5] we believe that this approach is not useful for our question answering and spatial inference task. Our system only accepts verbs like *to be* and *stand* that contain no actions. The other three word types (nouns, adjectives and prepositions) are sufficient to describe static scenes when combined with these verbs.

The part of speech (POS) tagger and parser from CoreNLP is used to extract POS tags for the given piece of text. The tagger outputs Penn Treebank POS tags [11] for each word. Our system has defined procedures for each POS category that are run in a particular order to generate a scene. The following sections discuss how each part of speech is handled by our system.

Nouns In our system, nouns correspond to 3D models from the Alice gallery. It tries to come up with an accurate model for the noun when it encounters it in a sentence. Although this sounds trivial, processing nouns requires many computational steps which may not be obvious at first. The following are some of the problems in representing nouns visually:

– A noun may refer to a non-physical entity.
– A noun may refer to a physical entity but it might be too general.
– A physical entity can be referred to in multiple ways.
– There is not always a one-to-one correspondence with a noun and a 3D model but sometimes a noun phrase can refer to it. The opposite is true as well. A noun can refer to more than one object.
– A noun can be ambiguous. It is impossible to get the word sense by just using the word itself without the context.

For our language visualisation system, we are not concerned with abstract or conceptual nouns. We are only interested in physical entities which can be seen and touched that exist in a physical world. These constraints remove many nouns in language from our candidate set because they cannot be visualized. It is easier to agree on a depiction of a physical entity, however this is not the case for abstract concepts like *democracy*.

For a system that needs to pick accurate models for words, it must have a mapping between these two. However listing all physical entities in language and matching them with 3D models appears to be infeasible. The main reasons for this is that there are many words in language whereas 3D model libraries are limited in size and nouns do not have one-to-one relationships with them. The relationship can be one-to-many and many-to-one. There is a need for another layer between these two lists of models and words. WordNet helps us to form this middle layer.

Every word in WordNet belongs to a synset (synonym set) and words in a synset have the same sense [9]. We matched our 3D models to the synsets that describe them best. Therefore we formed a relationship between a word and its synset and then by using that synset the system can reach the model for that word. A synset can also be matched by more than one model in a library. For example, there does not need to be only one *cat* model in our object database. This mapping is done manually for the 528 models in the Alice object gallery. As a synset consists of many different words, our system can handle a much larger number of nouns.

For the representation of nouns, we also benefit from the hyponym and hypernym relationships (is-a or kind-of relationship) of WordNet. A noun in WordNet is positioned in a tree structure where the root node is "entity". Every noun is a subclass of that root node which means that every noun is an entity. Nouns become more detailed and specific deeper in the tree structure. This kind of hierarchical relation between words gives useful information like "A cat is an animal" and "A car is a vehicle".

For all nouns we know their parent categories. This knowledge enables us to show a model for a specific type of noun when a user enters a general name instead. When the user asks for an animal it is perfectly acceptable and reasonable for our system to show a *bird* or a *cat* or a *dog* or any other it finds as a child of the *animal* noun. This requires us to match our 3D models to the synset of the deepest noun we can find in the tree. By doing that we can fetch the model that the user typed specifically or we can again find a suitable model for a more general word.

In our system, we do not disambiguate word senses, but we use the first word sense (or synset) that is associated with that word. The first word sense in the WordNet is the most commonly used sense of the word. In future work, word sense disambiguation can be integrated and this can improve the overall accuracy of natural language understanding.

Adjectives In grammar, adjectives are the describing words for nouns and noun phrases. They give more information about the object signified. In our system, we use the CoreNLP parser to parse natural language sentences and it gives us the modifier(s) of each noun in the sentence.

One important observation about adjectives in language is that many of them cannot be visualized in a 3D scene. In our language visualization system, every 3D model is associated with nouns and these models have six different properties which are visibility, size, position, orientation, color and transparency. We chose three of these properties (size, color and transparency) that can be modified by adjectives. The adjectives that modify these three properties can be called depictable or visualizable adjectives. We can also refer to them as model modifying adjectives since they have a visual effect on the appearance of the 3D model in a 3D scene.

For instance *jealous* is a type of adjective that does not change one of the three properties listed above and therefore it is not visualizable in our system. Our system knows which adjectives have a visual effect we can control and does the necessary modifications for the model. If it is not depictable, the system just skips that modifier. However, a *jealous girl* model can be present that is separate from the *girl* model. It should be picked and shown if *jealous* is not a depictable. The sequence is important here. The system first searches for a corresponding model for the 'adjective + noun' pair. If a model for that pair is found, that would be a more accurate result and system would show that. If the adjective is a model-modifying type, the system modifies the model accordingly. If not, it ignores the adjective and just retrieves the corresponding model for that noun.

For the spatial inference task we only focused on depictable adjectives related to size. The other two properties of a model can also be changed manually, but it is not automated yet. In the future versions, that can be implemented as well. The adjectives related to size can modify one or more of the three size related properties which are width, height and depth. As adjectives modify nouns in a language, in a language visualisation system they correspond to procedures that are applied on models. Similar to what we did about nouns before, for these kind of adjectives, we did a matching between their WordNet synsets and corresponding procedures. By doing this, we associated adjectives that have the same word sense with the same procedure. The user sees the same effect on the scene no matter which word she picks.

WordNet does not have hypernym or hyponym relations for adjectives since it is not very meaningful for adjectives to have hierarchical relations. However another important information extracted from WordNet is the attribute information of an adjective. For example, for the adjective *tall*, WordNet tells that it modifies the *height* attribute or for *fat* it returns *fatness.* These attributes of the adjectives are matched to the attributes of the 3D models and when the system encounters any of them it modifies the related attribute accordingly by using a scalar value for the degree of the adjective.

Prepositions Prepositions are the word types which denote temporal and spatial relations. They have complements which they relate to the context they occur in. For our spatial inference task we limited ourselves to spatial relations. These are the prepositions that denote location and direction. Since our system only handles static verbs direction related prepositions are not covered. Location related spatial prepositions like *in front of, behind, near, next to, beside, above, below, on, in* are mapped to procedures in our system. These procedures take two arguments where one of them is a landmark object which the other object is placed relative to. The object is placed by translating it in the corresponding direction denoted by the preposition. The translation amount is determined by the size of the objects. For prepositions like *near*, multiple directions are possible and in that case the direction is chosen randomly. First these procedures try to place an object according to the spatial relation while keeping the position of the other object fixed. If these objects already have other spatial constraints on them, then they are placed without breaking the previously applied constraints. This is done by reversing the spatial relation and the landmark object or by moving the other objects that are part of the previous spatial relations. Prepositions are not included in WordNet, so a mapping from the synsets to procedures is not possible. We directly mapped words to related procedures in this case.

This section summarized how each POS contributes to our system. In the next section, we will discuss scene construction which makes use of these to construct scenes from natural language sentences.

4.2 Scene Construction

After an empty scene is presented, the user can directly enter a piece of text that describes a scene with multiple sentences or she may choose to build the scene incrementally providing one sentence at a time. In either case, after the text entering is complete, the system splits sentences (if there is more than one) using CoreNLP and parses each sentence. For each sentence, nouns are extracted and the system searches WordNet for the synset of the the target word. If it is

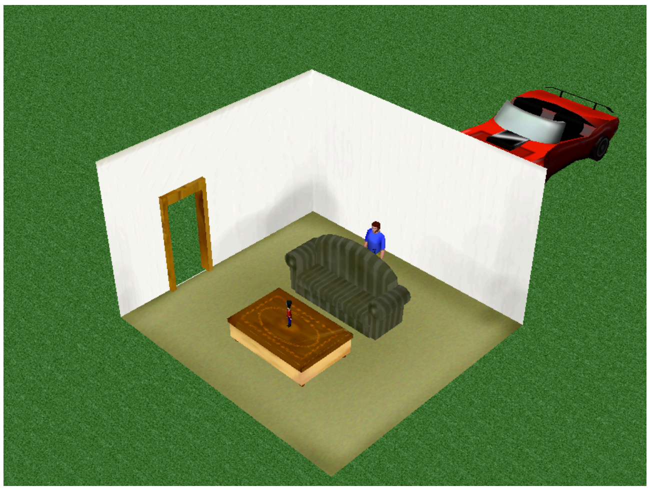

Fig. 1. There is a room. A sofa is in the room. A table is in front of the sofa. A toy is on the table. A man is behind the sofa. A car is behind the room.

found, system than searches for a 3D model in our synset-model mapping data source.

If there is not a suitable model for that synset, the system switches to the children of the synset using hyponym relations in the hierarchy of WordNet. If there is more than one model for that particular synset, the system picks a model randomly. If there is no exact match, to come up with a similar model for the noun, our system asks user whether to search coordinate terms or not. These are the nodes that share a hypernym in the WordNet hierarchy. It asks the user because the result may not be always close to what the user has in mind. For example when the user asks for *cheese* the system cannot find it and comes up with *banana* which are both children of *food*. After these efforts if a suitable match is not found, the system just skips that noun. For instance this is the case for abstract concepts.

When a model is picked succesfully, all of its modifiers (adjectives) are extracted from the parser output and the necessary modifications are made to it. As described in the previous section, these modifications are related to size and based on the modifier width, height and depth properties of the models are changed. A mapping is done between the attributes that the adjective modifies and a procedure that applies that modification to the model. If the modifier of the noun is found in that mapping than the procedure is applied to that model, if it is not found the system just skips that modifier.

After this stage, our system extracts spatial relations (prepositions) from the sentence. Models are placed into their correct places to satisfy the constraints in the sentences. As described in the previous section about prepositions, a basic constraint resolution system is used to satisfy the constraints. Constraint resolution is necessary as the number of models increases in the scene. In addition to that, we expect the user not to enter any conflicting constraints. In that case, system can satisfy some of them and not produce satisfactory results.

A fully constructed scene can be seen in Figure 1 above. After the construction is complete, the user may go on with the question answering, which will be described in the next section.

4.3 Question Answering

After the scene construction is completed, the user may switch to question answering mode and ask questions about the current scene. At any point she may choose to further describe the scene and then continue with question answering. A question is asked in natural language text and based on the keywords in the question it is categorized into a specific question type.

The system can handle three types of questions. First of all, user can directly ask about the position of an object. The system answers this question relative to other objects (landmarks) in the scene. It knows which prepositions correspond to which spatial relations and our rule based sentence builder turns these relations into sentences. It iterates over each spatial constraint in the scene and checks whether the object in question is part of that relation. If it is, then these relations are converted to natural language in the form of *X is in relation to Y*.

Secondly, another type of question a user may ask is a yes/no question. The user may test if a certain spatial relation is present between two objects. Again the system converts prepositions in the questions to the geometric relations and checks if the target object and landmark object satisfy those criteria. Finally, a user can ask whether two objects can see each other or not. For this case, the system first considers orientation, it checks whether the tested object is in front of the other object or not. If it is not in front of the other object then it cannot be seen. If it satisfies this criteria then we draw an imaginary beam from the source object towards the target object. If it is intersected by another object in the scene, the system concludes that two objects cannot see each other. Even in very complex scenes, it is an easy task for our system to come up with the correct answers for these kinds of questions. Some sample questions and answers are shown in Table 1 below.

Table 1. Question answering examples for the scene in Figure 1

Questions	Answers
Where is the room?	It is in front of the car.
Where is the sofa?	It is in the room, in front of the man, behind the table.
Is toy on the table?	Yes.
Is man in the room?	Yes.
Is car in the room?	No.
Is sofa in front of the table?	No
Can man see the sofa?	Yes.
Can man see the car?	No.

5 Conclusion & Future Work

In this work we presented a novel natural language to 3D scene conversion system. It uses an extensive set of 3D models and a rich vocabulary to construct a scene. The scene can be modified and the state of it can be queried using natural language. We have shown that how it can be used to solve some types of spatial inference problems. The current state of our language visualisation system is limited yet promising. As part of the future work, we plan to integrate actions via dynamic verbs. This can be done extending our rule based engine with path planning algorithms, a physics engine and animations. This will allow us to create more complex scenes and cover an important part of the language. This will introduce the time component to our system, as these verbs will form animations and animations will span a fraction of time. Question answering will become much more sophisticated, as it should investigate many scenes and consider the time sequence of events. Although real world and physical interactions are a huge part of the language, they are not all of it. The framework described here is not capable of understanding ideas, feelings and abstract concepts in general. We believe that these issues can be addressed with engines similar to our work here for the physical world that can fill the gap between language and these concepts.

References

1. Clay, Sharon Rose, and Jane Wilhelms. "Put: Language-based interactive manipulation of objects." Computer Graphics and Applications, IEEE 16.2 (1996): 31-39.
2. T. Winograd. Understanding Natural Language. PhD thesis, Massachusetts Institute of Technology, 1972.
3. Dupuy, Sylvain, et al. "Generating a 3D simulation of a car accident from a written description in natural language: The Carsim system." Proceedings of the workshop on Temporal and spatial information processing-Volume 13. Association for Computational Linguistics, 2001.
4. Ma, Minhua. Automatic conversion of natural language to 3D animation. Diss. University of Ulster, 2006.
5. Coyne, Bob, and Richard Sproat. "WordsEye: an automatic text-to-scene conversion system." Proceedings of the 28th annual conference on Computer graphics and interactive techniques. ACM, 2001.
6. Barbu, Andrei, et al. "Simultaneous object detection, tracking, and event recognition." arXiv preprint arXiv:1204.2741 (2012).
7. Roy, Deb K. "Learning visually grounded words and syntax for a scene description task." Computer Speech & Language 16.3 (2002): 353-385.
8. Alice, `http://www.alice.org`
9. Miller, George A. "Five papers on WordNet. " Technical Report CLS-Rep-43, Cognitive Science Laboratory, Princeton University (1993).
10. Stanford CoreNLP, `http://nlp.stanford.edu/software/corenlp.shtml`
11. Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Treebank-3. Linguistic Data Consortium, Philadelphia.