International Journal of Scientific & Engineering Research Volume 11, Issue 4, April-2020
ISSN 2229-5518

Constraint Based Interior designing interface for scene generation using text

Yashaswini S
Assistant Professor
Department of Computer Science and Engineering
Cambridge Institute of Technology
Bangalore, India

Shylaja S Sharath
Chairperson,
Department of Computer Science and Engineering
PES University
Bangalore, India

Abstract— **spatial constraint consideration while retrieving and grounding the appropriate object is a complex task in text to scene generation. When the user describes the scene in the natural language, the system has to decipher the context of the user description and retrieve related 3D models. Natural language not always explicitly specifies all the constraints. The system has to understand the hidden implicit constraints. This work mainly concentrates on making the system learn all the implicit constraints and satisfy the user with the output generated for given input description.**

The system should learn the basic common sense knowledge using a Machine Learning algorithm called Long Short Term Memory (LSTM). This algorithm takes text corpus as input which contains all the user defined descriptions including the implicit constraints. It helps in training the model to predict the implicit constraints given a natural language description as input.

I. INTRODUCTION

Text to 3D scene generation system does the task of analyzing the input text and giving a 3D scene as output. This 3D scene involves the grounding of appropriate 3D models from the user input text description. The natural language text description does not include all the constraints. Few will be explicitly specified and rest has to be implicitly understood by the system. This is the basic common sense knowledge that the system has to learn.

Lexical grounding is nothing but retrieving the 3D models and placing them in appropriate position. This work concentrates on the sub problem of lexical grounding. That is, identifying the implicit constraints. Finding solution to this sub problem is very important since it has major impact on quality of scene generation. It also affects the further stages which involve correctly arranging the retrieved 3D models.

The most basic approach to retrieve 3D models would be keyword search. But this approach fails to generalize the description well and does not consider the implicit constraints which are necessary, hence machine learning algorithm called Long Short Term Memory (LSTM) is used to make the system learn and identify the implicit constraints.

LSTM networks are special kind of RNN (Recurrent Neural Network) capable of learning long-term dependencies. Recollecting data for long period of time is their default behavior, not something they battle to learn. LSTMs help preserve the error that can be back propagated through time and layers. By maintaining a more constant error, they allow recurrent nets to continue to learn over many time steps (over 1000), thereby opening a channel to link causes and effects remotely.

E.g. : Consider an input " cake is placed on the table". Now the model should identify what all models to retrieve. i.e. a cake, plate, table, chair

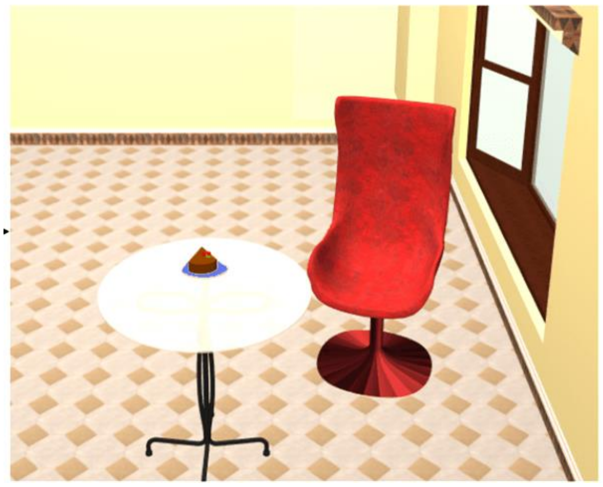


Figure 1: Implicit constraint [1]

This work has made use of ShapeNetSem database. ShapeNetSem is a dataset of 3D models which are highly annotated with physical attributes. ShapeNetSem is a smaller, more densely annotated subset of ShapeNet consisting of 12,000 models spread over a broader set of 270 categories.

II. Problem Formulation

A. Identifying the implicit constraints

This work strives at implementing an effective methodology that helps the system to retrieve appropriate 3D models w.r.t. given input description. Taking the sub problem of lexical grounding into consideration, i.e. identifying the implicit constraints, we construct a machine learning model which does the task of predicting unspecified necessary objects. This work also takes into account the object count if specified.

III. The Methodology

Artificial neural techniques like Deep learning are used to learn high dimensional, non-linear huge datasets, which is very complicated network structure with many hidden layers. Sequential layer is a stack of layers. We can add any number of layers using the add () function. The input shape has to be mentioned in the first layer. Other layers can do shape inference from the first layer.

Input to LSTM, has to be 3 dimensional, which is specified in the first hidden layer of the network. It is specified in the 'input_shape' argument in the LSTM layer. So when fitting the model, or when making predictions, 3D array has to be passed. 'input_shape' takes 3 arguments (samples, timesteps, features).None takes any positive integer as input. (Batch dimension is not included in this function) By default the 1st argument expects to pass 1 or more samples. One sequence indicates one sample. Batch can have many samples. Batch size is the first argument.

The size of the training text has to be known for defining the word embedding layer. The next layer is one hidden LSTM layer, followed by one dense output layer. The activation function is required because without it the output signal simply is a linear function. The linear function cannot learn much of the complex mappings between in the data.

Without activation function, the model would just be a linear regression model. Now it can learn from complex data like images, audio and video. The activation function should be differentiable to perform the back propagation to calculate the error, i.e the loss and update the weights according to the extension of gradient descent algorithm, Adam optimizer.

There is single learning rate for all weight updates and it does not change during training in stochastic gradient. In Adam optimizer, learning rate is maintained for each network weight. Relu Activation function is used only for hidden layers. So for output layers, Softmax activation function is used, for a classification problem to calculate the probabilities for the classes.

Word embedding layer which takes the total size of the training text, 10 dimensional projection, where each word in the corpus is specified by a real valued vector of size 10.input_length here, has the size of the sequence of maximum length. The vocabulary has a real valued vector for each word in the corpus, where each vector has a specified length.The input size is taken as length of the maximum vector from all the inputs sequences.

Using Tokenizer class, word encoding is done. Mapping is done between the words and integers using the available functions in Keras. Word_index is used to fetch the size of the vocabulary. Function to_categorical() turns the labels, i.e. y vector into one hot encoding labels, of number of classes as the size of the text corpus. Compile and fit the network on encoded text data. Compilation is done before training the model. Categorical cross entropy loss function shows the loss between two probability distributions.

To train the model, fit function is used. Models are trained on Numpy arrays of input data and labels in Keras. Function to generate sequences and reshape the input data. The model which is to be trained is ‘model’, ‘tokenizer’, to convert input text to integer sequences. Padding is done to specify the fixed length, with ‘max_length’ argument. The probabilities of each word to the classes specified in Y array are predicted. This is a multi-classification problem where the numbers of labels are the number of words in Y array. The seed sentence is broken into words and it is searched in X array. When the match is found the break out of the for loop and append the corresponding Y label. List of predicted words is returned.

The model is trained with the help of AWS server for 400 epoches. This is done by the fit function where we specify the number of epochs, X and Y. After the model is trained we save the .hdf5 file with the least loss. When the predictions are made this file is loaded to apply the updated weights and make predictions. Output of the generate function is written to a file.

The text is read and tokenized into words using nltk tokenizer called RegexpTokenizer. Convert the tokens into lowercase using lower ( ) function. Stop words are the unwanted words apart from nouns, adjectives which are not needed. They are removed using words () function from nltk corpus. Duplicate words are removed in the list.

To identify the words as nouns, adjectives and numbers POS tagging is done using default tagger of Treebank Corpus which is available in nltk. Also, to consider the Count of objects, in token ( ) function, logic is written to check if a number, which is tagged as ‘CD’, is followed by a noun tagged as ‘NN’. At the end of output of LSTM model and the preprocessing 2D array is created, of size number of nouns in the seed sentence X 2. The second column has the number of count of objects, specified in the seed sentence.

The resulting output is processed. That is the stopwords are removed. POS tagging is done in order to retrieve the nouns and numbers. The resulting objects are mapped with metadata.csv file obtained along with ShapeNetSem dataset to obtain the object Id. This file has the list of all 3D model tags along with their Id. The object ID, along with the set position is used is used he's used in blender script. The code for rendering the models in the blender is called using sub process call.

IV. RESILTS & DISCUSSIONS

In this work we have mainly concentrated on retrieving implicit and explicit 3D models given a seed sentence as input. This work can also handle input cases where the user asks for specified number of objects.

As the spatial constraints are not considered, the 3D models are grounded in random manner. It has not considered the size of objects, its position and the co-relation between the objects.

Below are the scenes that are generated for the given input text. It is rated on the scale of 1 – 10 by human evaluation.

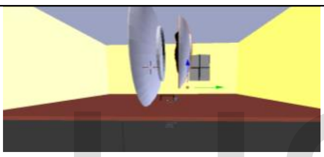
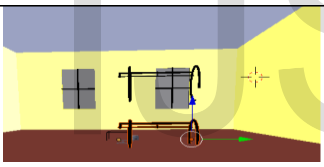
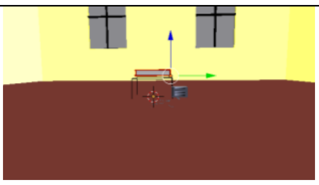
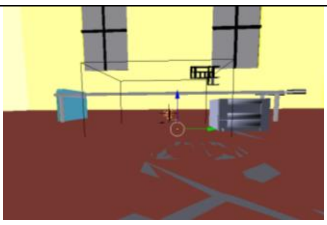
Test Id	Input Text	Scene generated
01	Cookies	
02	2 Chairs	
03	Book is on the table	
04	Luggage is on the bed	

Table 1: Result table consisting of input text, Generated scene and the human evaluated result

CONCLUSIONS

This work mainly focuses on getting appropriate 3D models when user inputs a seed sentence. In a natural language, not all constraints can be explicitly specified. Few of them should be understood using common sense knowledge.

In this work we are training a LSTM model with text corpus. The text corpus contains user-defined descriptions of scenes. The description includes all the explicit and implicit constraints. The trained model takes a seed sentence as input and gives all the related sentences to it and the sentences, which comes next in sequence.

This output is then pre-processed to get all the objects. By human judgment it is understood that the model is not accurate enough to produce all the implicit constraints. So, using Concept-Net which is a common sense reasoning semantic network which helps in this work to give the implicit constraints which the trained model couldn't output.

After getting to know the output objects we now retrieve the object Ids from the metadata.csv file given by ShapeNetSem dataset. This file has also given the approximate location of objects. Knowing the object Ids and where it has to be placed, we can now get the 3D models and output them using Blender. This work handles the user input with number constraints. It generates right 3D models. Since the spatial constraint is not considered here. So the objects are placed in random dimensions.

FUTURE DIRECTIONS

The following features can be further added on to the current work.

- Considering adjective occurring before noun. Eg : Wooden Chair
- Considering the object size
- Identifying the parent and child object. i.e. placing objects w.r.t. each other

REFERENCES

[1] Angel Chang , Will Monroe , Manolis Savva,Christopher Potts and Christopher D. Manning “Text to 3D Scene Generation with Rich Lexical Grounding” Stanford University, Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL) and the 7th International Joint Conference on Natural Language Processing(IJCNLP), Held at Beijing, China, 26-31 July 2015, Volume 1,P15-1006

[2] Angel Chang, Will Monroe, Manolis Savva, Christopher Potts and Christopher D. Manning “Learning Spatial Knowledge for Text to 3D Scene Generation” .In Proceedings Of Empirical Methods in Natural Language Processing(EMNLP), held on 25-29 October 2014 in Doha, Qatar, pp 101

[3] Angel Chang, Manolis Savva, Christopher Potts and Christopher D. Manning “Semantic Parsing for Text to 3D Scene Generation” Stanford University, Conference: Proceedings of the ACL 2014.