Causal Scene Generation

Harish Ramani

Abstract

The project aims to procedurally generate 2D images from natural language where the causal model describes the relationship between the entities in the caption. By taking the vision- as – inverse graphics view of computer vision, we could think of captions as natural language representation of the data generating process that generated the accompanying image.

1 Introduction

A causal model aims to represent a data generating process. There have been numerous attempts to model text to an image and it predominantly involves usage of a Generative Adversarial Network or GAN. [1] shows an existing GAN model which attempts to display an image that describes the text, however ineffectively. It doesn't capture the necessary relation between all the entities in the image. This is where a causal model would be highly beneficial, to capture the relationship between the different entities. To give an example, we could have a caption "*A man is walking towards a door*". If we have a causal model and a procedural image generation scheme, we could ask a counterfactual question, "*Had it been a woman*", what would the image look like? This independent study is a humble attempt at figuring out the necessary pieces needed to solve scene generation from text by showing a proof of concept that works for a pre-defined universe with limited entities.

2 Background

The current work is an extension of a project titled "*Causal Object Oriented Programming*" where, a proof-of-concept was created to answer probabilistic queries for different entities that are related using Pyro [2] This work utilized Avi Pfeffer's concept of Bayesian Object Oriented Programming[3] which was originally created in a programming language named Figaro using Scala. This work allowed us to represent real world entities and their relationship with other entities, probabilistically with the familiarity of Object-Oriented Programming. In this work, we considered a toy example found in [3] and reproduced the results in pyro. The example and the corresponding code base for this work could be found here [4]

3 Methods

The different components required to generate a scene from caption are, a causal model that understands the relations between the entities, a procedural generation scheme to draw the 2D image given the necessary metadata about the image and a natural language parser to get all the required metadata from the captions.

3.1 Procedural Generation

A procedural generation is an automated way of creating data. Here, we are trying to automatically generate 2D images given some metadata about the image. In our example, we considered a universe of game characters (currently two characters Satyr and Golem) doing a limited set of actions like Attack, Taunt, Walk etc. Each set of character has different types where the armor, head color and the way they perform the actions varies. The procedural generation code creates different versions of the character which didn't exist in the original set.

HEAD HAIR TORSO WEAPON GENERATED CHARACTER

The above is an example of how a 2D image can be generated from different metadata. With the help of the above rendering mechanism, we can generate a scene by giving some information about the scene. We basically use a look up table to get to the corresponding images and merge them all together. Each image is of the same height and width.

3.2 Causal Model

A causal model represents the data generating process. For scene generation, the interplay between the different entities in a scene can be captured by a causal model. To give an example, say, we have a caption "*A boy is walking on the road*". Here, the boy is an entity and the action that entity is doing is walking. The background environment where this scene takes place is the road. Suppose, if we ask a counterfactual question, "*Had it been a girl?* ", then we need to know about the data generating process and get the action and the background the girl (entity) has the high probability of occurring with. We use the pyro framework to create causal models.

3.3 Image Classification

Image classification refers to the labeling of images into one of a number of predefined classes.
There are potentially *n* number of classes in which a given image can be classified. A multi label image classification is one in which each image can belong to multiple classes at the same time. For our problem, we train a multi label image classification model where we give an image and expect the image to classify the character, its type and its action. Since our image has 2 characters in it, we expect 6 labels from each model prediction. The character of the actor, its type and action, the character of the reactor, its type and the reaction. We need this image classification model, so that we can condition on them in our causal model and generate counterfactual images.

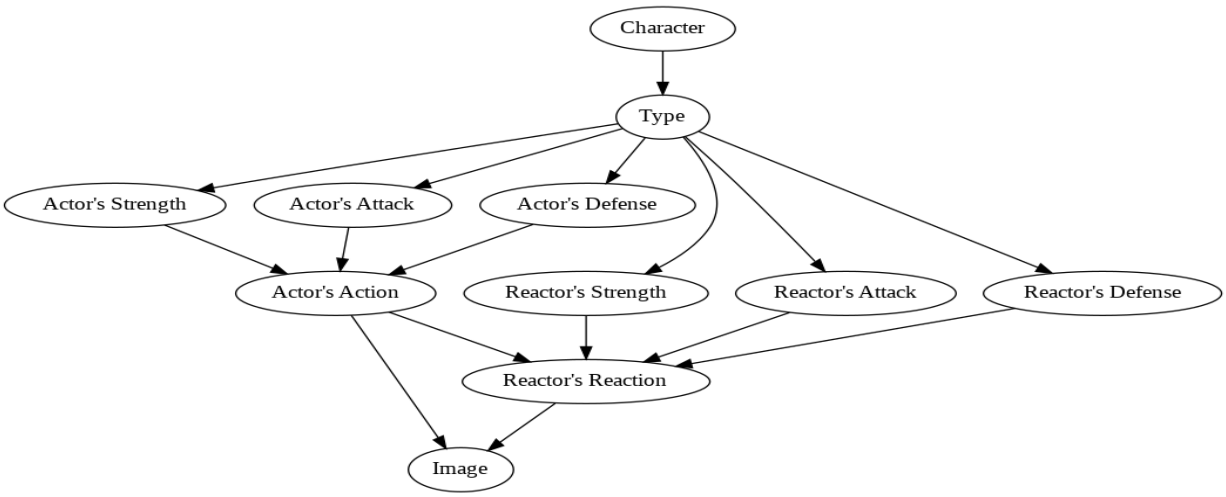
4 Implementation

To build a causal model, we decided to use the characters that appear in games. We chose two characters, Satyr and Golem and decided to give attributes like Strength, Defense and Attack that define their behavior. Each attribute can be either high or low. The causal model tries to represent what happens when these characters interact with each other. To model this, we first need the sample space of all the actions that they can do. The different activities that each character does are Walking, Taunting, Being Idle, Attacking, Getting Hurt if they're attacked and possibly die. Each Character has 3 different ways by which they can visually appear and they affect the character's attributes. All the different actions and the types that each character does correspond to a different image.

To make things simpler, each interaction has an actor, one who instigates or makes an action towards the other character based on his own attributes. The other character simply reacts to the action based on his own attributes. The probabilities in which each character is chosen and doing other actions are randomly assigned, for now, in the form of conditional probability tables and not learnt from data.

4.1 DAG

A Directed Acyclic Graph explains how the random variables in the process affect each other. In the following image, the character and its type affect the attributes, like strength, attack and defense. The actor's action in that scene depends on his strength, defense and attack. The reactor's reaction depends on his attributes and the actor's action. The image is basically capturing the actor's action and the reactor's reaction and hence is dependent on that.



4.2 Sample Image

We randomly sample the probabilistic model to get an actor's action and reactor's action along with the respective character and type. Based on this metadata, we can draw an image of the scene.

Fig1: Golem is attacking Satyr and it got hurt in the process.

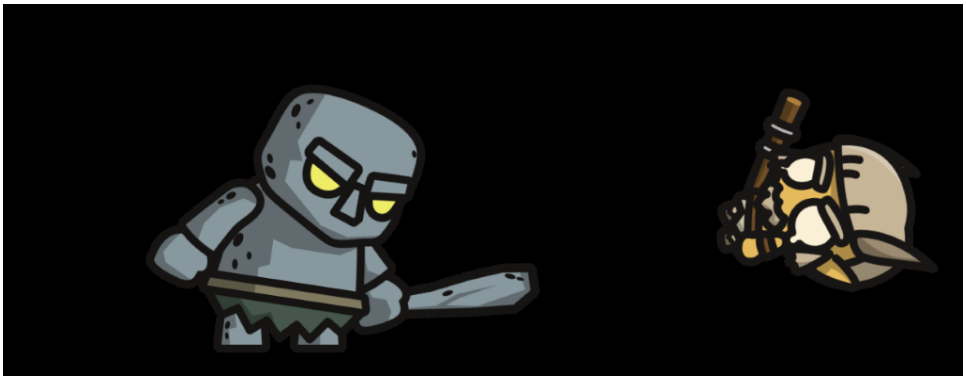


Fig2: Golem is attacking Satyr and it died in the process

4.3 Image Classification Model

The purpose of an image classification model is to train a model to label the character, type and action of both the actor and reactor when given an image. This would be used by our causal model to condition on the image. Currently, Image classification problem is best solved by the use of a Deep Learning Framework. We set up a multi layered network and train using the Adam optimizer to label the image.

4.3.1 Data Pre-Processing and Data Augmentation

Using our procedural image generation scheme, we produce all the different possible combinations of images (2 characters with 3 types each and the actor having 3 actions and the reactor having 4 reactions to choose from) We produce a total of 432 different images. Since a neural network needs a lot of training examples to label effectively, we use different data augmentation techniques to increase the sample size. The different data augmentation techniques include, rotation of the image, flipping the image left to right, flipping the image top-down and adding gaussian noise. Thus, we add 4 variations to each image. The data pre-processing step includes resizing the image to 400*400 pixels, converting to grey-scale and normalizing the RGB values by dividing each pixel by 255

4.3.2 Model Architecture

We use a sequence model with multiple convolutional hidden layers with max-pooling and dropout units. The convolutional units convolve the inputs and with the help of a learning algorithm learns the inherent features that differentiates an image from other. The max pooling layer captures the most present feature in that image and dropout layer acts as a regularization scheme to keep the weights bounded to a low value.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 396, 396, 16)	1216
max_pooling2d_1 (MaxPooling2	(None, 198, 198, 16)	0
dropout_1 (Dropout)	(None, 198, 198, 16)	0
conv2d_2 (Conv2D)	(None, 189, 189, 32)	51232
max_pooling2d_2 (MaxPooling2	(None, 94, 94, 32)	0
dropout_2 (Dropout)	(None, 94, 94, 32)	0
conv2d_3 (Conv2D)	(None, 85, 85, 64)	204864
max_pooling2d_3 (MaxPooling2	(None, 42, 42, 64)	0
dropout_3 (Dropout)	(None, 42, 42, 64)	0
conv2d_4 (Conv2D)	(None, 38, 38, 64)	102464
max_pooling2d_4 (MaxPooling2	(None, 19, 19, 64)	0
dropout_4 (Dropout)	(None, 19, 19, 64)	0
flatten_1 (Flatten)	(None, 23104)	0
dense_1 (Dense)	(None, 128)	2957440
dropout_5 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8256
dropout_6 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 23)	1495

Total params: 3,326,967
Trainable params: 3,326,967
Non-trainable params: 0

4.3.3 Model Approaches

Two different model approaches were considered. In the first approach, both the actors and reactors were present in the image and the model tries to classify the character, type and action of both the actor and the reactor. In the second approach, the images were split into two depicting either the actor or the reactor in each image. The model then tries to classify the entity that is appearing (either actor/reactor), the character and the type and their action.

5 Results

5.1 Image classification Model

Approach	Training Accuracy	Validation Accuracy
Actor and Reactor in same image	0.8287 (N=1725)	0.8563 (N=69)
Actor and Reactor split into different images	0.8753 (N=691)	0.9538 (N=173)

For the second model, we didn't perform data augmentation and hence the number of samples were considerably less than the first model but the training accuracy and the validation accuracy surpass the first model.

6 Discussion

TODO

References

[1] https://experiments.runwayml.com/generative_engine/

[2] http://pyro.ai/examples/intro_part_i.html

[3] https://www.manning.com/books/practical-probabilistic-programming

[4] https://github.com/robertness/causalML/tree/master/projects/causal%20OOP/causal_oop_social_media/ppp_replication