Dataset

Preparation:

Image

- Encode all the objects
- Encode all the categories
- A chart specifies the category and sub-categories that an object belongs to
- Each layer in the SVG file should be labeled by the corresponding Object ID
- Occlusion can be inferred from the order of the layers automatically

Text

- Encode all the images
- A text description associated with each image, labeled by the image ID

Categories:

0 Person:

0.0 Lifestyle, 0.1 Business, 0.2 Technology, 0.3 Festival

1 Surrounding

1.0 Indoor

1.0.0 Office

1.0.0.0 Office, 1.0.0.1 Hall

1.1 Outdoor

1.1.0 City

1.1.0.0 Street

1.1.1 Wild

1.1.1.0 Park, 1.1.1.1 Forest, 1.1.1.2 River

2 Background

3 Decoration

Caveats:

- When there's only one layer, it must be the surrounding layer, with white background
- Background layer is always in the most back.
- Decoration layer is always in the most front.
- Object, especially person, seldom gets re-used in the image.

Baseline I: Conditioned discriminator + brutal-force search predictor

Task: Automatically generate an image given input text using discriminator and brutal-force search

Feature engineering:

Image embedding: (1 + 4 + 6 + 3 + 6 + 2 ~~+ N_OBJECT~~)

- Number of layers: [1-4]

- Categories (Layers): N_CAT (4) (Binary)
- Sub-categories: N_1ST_CAT (6) (One-hot)
  N_2ND_CAT (3) (One-hot)
  N_3RD_CAT (6) (One-hot)
- Occlusion: **Person** in front of **Surrounding** or otherwise [0,1]
- ~~Object: N_OBJ (Binary) (Touch object level later)~~

Text embedding:

- **Tokenization**: Treebank word tokenizer (NLTK standard tokenizer)
- ~~**Spell correction:** (later)~~
- **POS tagger:** Penn Treebank tag set (NLTK standard POS tagger)
- **Lemmatization**: WordNet lemmatizer
- ~~**Coreference resolution:** Stanford CoreNLP (later)~~
- **Similarity replacement (unseen words):** WordNet Leacock-Chodorow Similarity (The top word in vocabulary with similarity > 2.6) (Or pre-trained word2vec)
- **Ngram:** (1-3)
- **Vectorization:** TF-IDF

- TF-IDF vectorizer on the entire set of text descriptions? (vocabulary size should be small. Sentence should be simple.)
- Semantic tagging, extract nouns and predicate tuples. (ICCV2013) Detect and eliminate cooccurred nouns. (Stanford CoreNLP) Then use count vectorizer.

Model - Discriminator:

Input: text, matching image
Output: consistent, inconsistent
Skeleton: Binary classification - Logistic regression
Architecture:

- ~~Sentence -> Word-level one-hot encoding -> bidirectional LSTM (or pretrained model) -> vector~~
- Corpus -> TF-IDF vectors
- Image -> feature
- Concatenation: text embedding + image feature
- Logistic regression

diff this relies on features associated text and image? Or we should use deep learning, but we don't have so much data, nor enough features

Training:

Input: A triplet of text-image pairs

- Text – image: consistent
- Text – mismatched (or randomly generated?) image: inconsistent
- Image – mismatched text: inconsistent

Loss: Add all three cross-entropy losses

Metric:

- L2 loss between image embedding?

- Precision/recall on the text-image pairs

Model – Predictor (Search agent):

Input: text

Model: text – all possible combinations of image features -> discriminator -> argmax

Combination pattern: (Choose one object arbitrarily in the specified category)

- **1 layer:** S
- **2 layers:** PS, PB, PD, SB, SD, BD?
- **3 layers:** PSB, PSD, PBD, SBD
- **4 layers:** PSBD
- **Occlusion: Person** in front of **Surrounding** or otherwise

Tuning

Evaluation:

- L2 loss between image embedding
- Precision/Recall ~~at object level~~, at category level
- Generate description from the predicted image, then compare to the original description use textual metrics (put aside for now)
- Human evaluation

Results
Error analysis
Optimization: Text embedding
Demo
Packaging

Model 2: Probabilistic model (ICCV 2013)
Model 3: Sequence model (CVPR 2019)
Model 4: Variational autoencoder
Model 5: Conditioned GAN (ICML 2016)