

Appendix

Table of Contents

| | |
|---|----------|
| A Further GeneCIS Details | 1 |
| A.1 Task construction | 1 |
| A.2 Implementation Details | 2 |
| A.3 Dataset noise | 2 |
| A.4 Discussion on symmetry | 2 |
| B Specific Solutions | 3 |
| C Results with ViT-B/16 on GeneCIS | 3 |
| D Combiner Architecture | 3 |
| E Further Implementation Details | 3 |
| F. Extended Related Work | 4 |
| G Qualitative Examples | 4 |
| G.1 GeneCIS examples | 4 |
| G.2 Model Predictions | 4 |
| G.3 Mined Triplets from CC3M | 4 |
| H Attributions | 4 |
| I. Acknowledgements | 4 |

We provide additional details and discussion of components of the main paper. We particularly highlight Appendix **A** for details on GeneCIS construction, and Appendix **G** for qualitative examples.

A. Further GeneCIS Details

Here, we provide details on the construction process of each GeneCIS task, making reference to the examples from Figure 2 for clarity.

A.1. Task construction

Focus on an Attribute: VAW [56] contains bounding box annotations for various objects, as well as a list of *positively labelled attributes* and *negatively labelled attributes* for each object. Note that, as discussed in § 3.1, it is impossible to exhaustively label an object for all possible ‘positive’ attributes. It is, however, possible to determine a set of ‘negative’ attributes. For instance, one *cannot* exhaustively label a ‘thick’ tree trunk as {‘wide’, ‘fat’, ‘large’...etc.}, but one *can* determine that it is not ‘thin’.

For this task, we construct templates by first sampling a reference object (e.g ‘laptop’) and identifying all *positive*

attributes of the object (e.g. ‘white’, ‘plastic’) and their corresponding *attribute type* (e.g. ‘color’, ‘material’). Given an attribute type, we select a ‘correct’ target image to have the same object category and attribute within the attribute type as the reference (a ‘laptop’ with the same ‘color’). Distractors are then mined to have the same object category but to be explicitly *negatively labelled* for the reference attribute (e.g. laptops which are negatively labelled for ‘white’). The condition in this case is the attribute type (‘color’).

Change an Attribute: We first select an anchor *attribute type* (e.g ‘color’), before choosing a reference image and a ‘correct’ target image which share the same object category, but have different attributes within the attribute type. In Figure 2, the reference and ‘correct’ target are both have the same object category (‘train’) but have different ‘colors’. The attribute of the ‘correct’ target is given as the condition (‘olive green’), and a model must understand the category of the reference image, as well as the attribute specified in the condition, to solve the problem.

We include two forms of ‘distractors’ in the gallery. The first form includes images with the conditioning attribute (‘olive green’), but with a different object category (e.g. ‘tent’). These images behave as distractors for models which retrieve based only on the the condition (we include 9 such images). We also include 5 images with the reference object category but without the conditioning attribute (e.g ‘trains’ which are ‘red’), behaving as distractors for models which only use the reference image content.

Focus on an Object: For tasks where the condition contains an object, we take images of cluttered scenes from the multi-object COCO dataset [40]. We use COCO Panoptic Segmentation [36] data which contains dense category labels for every pixel in the image.

We first select a reference image and identify all of its constituent object categories, ensuring at least 10 categories are present. Next, we construct a set of all images in the dataset with at least 6 objects in common with the reference – but do not contain *all* reference categories – and rank them based on the extent of their category intersection (\mathcal{I}_{Close}). We also construct a set of images with very *few* intersecting objects as \mathcal{I}_{Far} . We consider the set of object category IDs in an image as a ‘bag-of-words’ descriptor for the image scene, with images in \mathcal{I}_{Close} containing a ‘similar scene’ to the reference, and \mathcal{I}_{Far} representing a ‘different scene’.

We randomly select the ‘correct’ target image from \mathcal{I}_{Close} , and the *conditioning object* is selected as one of this image’s intersecting objects with the reference (e.g. ‘refrigerator’ in Figure 2). The first form of distractors is mined by taking images in \mathcal{I}_{Close} which *do not* have the conditioning object. These examples confuse models which only use the reference image (there are 9 of these). Another type of distractor is constructed by taking images from \mathcal{I}_{Far} which *do* have the conditioning object, confusing models which only

consider the text condition (there are 5 of these). In this way, only the target image has both a *similar scene* and also the *conditioning object*, and is thus *conditionally* the most similar image in the gallery. In Figure 2, only the target image contains a ‘refrigerator’ and is ‘outside’. We note that solutions which only match ‘bag-of-objects’ descriptors fail here (e.g. those which simply detect all objects in the images): the ‘correct’ gallery image is randomly selected from \mathcal{I}_{Close} and does not necessarily contain the highest object category overlap.

Change an Object: This task is constructed in a similar form to ‘Focus on an Object’, in that we first select a reference image and construct \mathcal{I}_{Close} and \mathcal{I}_{Far} . Differently, in this case, we first select the ‘correct’ gallery image from \mathcal{I}_{Close} as the most similar image which does not have perfect object overlap. Next, the conditioning object is selected randomly from the objects which *do* appear in the ‘correct’ gallery image, but *not* in the reference (‘ceiling’ in the example in Figure 2). Distractors are constructed from both \mathcal{I}_{Far} and \mathcal{I}_{Close} , such that they do, and do not, contain the conditioning object respectively. There are 5 distractors from \mathcal{I}_{Far} and 9 distractors from \mathcal{I}_{Close} .

A.2. Implementation Details

Attribute-based Tasks: A taxonomy of *attribute types* is provided in VAW [56], containing diverse attribute types from ‘letter color’ to ‘texture’. We manually clean and refine the taxonomy for our purposes, for instance reassigning many attributes which were assigned to the ‘other’ attribute type. The resulting taxonomy contains 45 attribute types with 663 constituent attributes. We build the tasks such that they are roughly balanced with respect to attribute type, noting that for some attributes it was not possible to construct a suitable retrieval template. For the ‘Focus on an Attribute’ task, we manually filter attribute types which do not form clear and visually grounded attribute categories. Specifically, we filter: ‘*opinion*’; ‘*other after*’; ‘*other physical quality*’; ‘*state*’; and ‘*type*’.

Finally, when cropping an object with a bounding box, we dilate the box by a factor of 0.7 in height and width, before padding the resultant image to square with zeroes. This allows some context to identify the object (we often found it difficult to categorize the image without this), and also maintains the aspect ratio of the underlying object. We chose the dilation factor which maximized the discrepancy between the ‘Image + Text’ and ‘Image Only’ Recall@1.

Object-based Tasks: The object-based datasets are derived from the validation set of COCO Panoptic [36], containing 57K images with 133 categories. The categories include ‘thing’ classes like ‘zebra’ and ‘bench’, as well as ‘stuff’ categories like ‘sand’ and ‘roof’. We only consider an object category to be present in an image if it occupies more than 1% of the image pixels. After a conditioning object is

selected, the COCO category name is given as a condition, and we strip miscellaneous identifiers such as ‘-stuff’ and ‘-other’ from the category names.

A.3. Dataset noise

Our tasks are built upon manual annotations in the VAW [56], COCO [40] and Visual Genome [37] datasets. These are widely used datasets in the vision community and, as such, our tasks should be error free in principle. However, we find some templates provide ill-posed problems through noise and ambiguities in the underlying annotations, as well as through bounding box dilation for attribute-based tasks.

Noise in the datasets is easy to understand, constituting instances where an object category or attribute is obviously mislabelled. However, the ambiguities are more subtle, and are artefacts of the underlying taxonomies of the datasets. For example, in some COCO images, ‘ceiling lights’ are labelled as ‘ceiling’ instead of ‘light’. This is not necessarily wrong, but reflects the fact that labels are defined in a one-hot manner and these pixels could refer to either object category. This is particularly difficult for attribute-based annotations, as interpretations of attributes are highly subjective (e.g. the definitions of ‘wide’ and ‘narrow’ are open to interpretation). We highlight that such label ambiguity, though underexplored, is present in almost all computer vision datasets, including in ImageNet [72].

We address this in a number of ways. Firstly, we ran a version of our method with 10 random seeds as well as on 10 cross-validation splits, finding the standard deviation in Recall@1 on each task to be around 0.2%. Although this does not quantify noise in the dataset, it gives an indication of what can be considered ‘signal’ on the tasks. Secondly, we find that the ‘Image + Text’ baseline outperforms the ‘Image Only’ and ‘Text Only’ baselines on most tasks, suggesting that the tasks measure conditional similarity. We discuss the exceptional case of ‘Focus on an Attribute’ in Appendix C. Thirdly, we evaluate at Recall@{1, 2, 3}, to account for any templates in which a ‘distractor’ image (*i.e.* ‘incorrect’ target) in the gallery actually constitutes a valid solution to the problem. Finally, we are in the process of manually filtering and verifying the templates, presenting the current version as ‘GeneCIS v0’.

A.4. Discussion on symmetry

We highlight that ‘similarity’, as discussed in this paper, does not describe a *symmetric* mathematical property. In GeneCIS, while the reference image is considered ‘similar’ to the correct target image given the condition, the reverse may not be true. For instance, in the ‘Change an Attribute’ example in Figure 2, the ‘green train’ in the reference is conditionally similar to the ‘olive green train’ target image, given the condition ‘olive green’. However, this target image is *not* similar to the reference image given the same con-

dition. In general, we find that ‘Focus’ tasks *are* symmetric given the conditions, but ‘Change’ tasks *are not*.

B. Specific Solutions

We design specific solutions for each of the proposed tasks in GeneCIS. These solutions take into account the specific construction mechanisms of each task and represent sensible approaches to tackling each task independently. We design all solutions to respect the ‘zero-shot’ nature of the evaluations and hence they are all based on ‘open-world’ models; we use CLIP [60] for the attribute-based tasks and Detic [81] for the object-based ones. All descriptions here refer to Figure 2 for clarity.

Focus on an Attribute: Given the attribute type in the condition (e.g. ‘color’), we first task CLIP with predicting the attribute of the reference image. Specifically, we use the taxonomy of attributes provided in VAW to construct a zero-shot classifier between attributes within that attribute type (e.g. {‘red’, ‘blue’, ‘white’} within ‘color’). Given the predicted attribute (e.g. ‘white’), we use its text embedding to find the nearest neighbour from the image embeddings of the gallery set.

Change an Attribute: We first use CLIP to predict the category of the the reference image, by constructing a zero-shot classifier from the categories in VAW. We then compute the text embedding of the concatenated predicted object name and conditioning attribute (e.g. ‘olive green train’) and find the nearest neighbour in the gallery.

Focus on an Object and Change an Object: We use the same specific solution for both of these settings. We first use Detic [81] to detect all object categories in the reference and gallery images, passing it the 2017 COCO Panoptic categories to construct the classifier. Next, we filter out any gallery images which *do not* contain the conditioning object category. Finally, using the detected object categories in a given image, we construct ‘bag-of-words’ descriptors of the reference image and the remaining gallery images. Specifically, these descriptors are binary vectors for each image (with elements for every COCO Panoptic category) and are set to ‘1’ if a given category is detected in the image. We use these descriptors to find the most conditionally similar image to the reference from the (filtered) gallery.

Discussion: Note that all of the solutions described here are specialized in two senses. Firstly, they are designed with the specific task construction method in mind, and hence are not applicable to all tasks as we desire for a general conditional similarity model. Secondly, all specific solutions leverage the underlying taxonomy of the *datasets* (VAW [56] and COCO Panoptic [36]) used in the benchmark.

C. Results with ViT-B/16 on GeneCIS

In Table 2, we report results on GeneCIS with a ResNet50×4 backbone for fair comparison with [3]. However, in Figure 6, we demonstrate that our model performs best when intialized with a ViT-B/16 CLIP backbone [18, 60]. We include results for this model in Table 6, along with the CLIP-only baselines described in § 6.1.

We first note that, with the ViT-B/16 backbone, our model outperforms all CLIP-only baselines, on all tasks and at all recalls. Particularly, with the ResNet50×4 backbone in Table 2, the ‘Image Only’ baseline outperformed ours on ‘Focus Attribute’ at higher recalls, which is no longer the case here. We further note that the ‘Focus Attribute’ task gives anomalous results when comparing the ‘Image Only’ baseline with ‘Image + Text’. Specifically, this is the only task for which the ‘Image + Text’ model does not outperform the other baselines. On this task, the information given by the condition is an attribute *type*, rather than the attribute itself (e.g. ‘color’ rather than ‘white’ in Figure 2). As such, the condition information likely only confuses existing vision models, and reduces the performance over the ‘Image Only’ baseline.

D. Combiner Architecture

The Combiner architecture takes in reference image and condition text features, composing them into a single vector as: $g(\mathbf{x}^R, \mathbf{e})$ where $g, \mathbf{x}^R, \mathbf{e} \in \mathbb{R}^D$.

The architecture consists of four functions (h_i , built from MLPs) which process features in parallel as:

$$g(\mathbf{x}^R, \mathbf{e}) = \lambda h_1(\mathbf{x}^R) + (1 - \lambda)h_2(\mathbf{e}) + h_3(\mathbf{x}^R, \mathbf{e}) \quad (2)$$

where $\lambda = h_4(\mathbf{x}^R, \mathbf{e})$ is a dynamic weighting of the features. We refer to [3] for full details.

E. Further Implementation Details

Our method: All models trained on CC3M were trained for 28K gradient steps. We train our strongest models with an initial learning rate of 1×10^{-6} and a cosine decay schedule, training both the CLIP backbone and the Combiner head with the same learning rate. We evaluate our model at each epoch, selecting the checkpoint with the best Recall@1 on the CIRR validation set [44]. Note that this single model is then taken and evaluated *zero-shot* on all benchmarks reported in this paper. Our optimizer is Adam [35] and we implement our models using PyTorch [52]. To fine-tune both the backbones and Combiner head with a batch size of 256, we train our models on 16 A100 GPUs, with a training time of approximately 12 hours. Finally, all features are normalized before the contrastive loss is computed.

Specific Solutions: For the attribute-based specific solutions, we use the same ResNet50×4 backbone as for our

Table 6. Evaluation on GeneCIS with a ViT-B/16 backbone where we evaluate our method and CLIP-only baselines. We find our method performs best with a ViT-B/16 backbone and that, with this architecture, our model outperforms the baselines at all recalls on all GeneCIS tasks.

| | Focus Attribute | | | Change Attribute | | | Focus Object | | | Change Object | | | Average R@1 |
|-----------------------|-----------------|-------------|-------------|------------------|-------------|-------------|--------------|-------------|-------------|---------------|-------------|-------------|-------------|
| | R@1 | R@ 2 | R@3 | R@1 | R@ 2 | R@3 | R@1 | R@ 2 | R@3 | R@1 | R@ 2 | R@3 | |
| Image Only | 18.1 | 30.1 | 40.6 | 11.5 | 21.9 | 30.9 | 9.4 | 17.0 | 25.4 | 7.6 | 17.1 | 25.5 | 11.7 |
| Text Only | 10.3 | 20.9 | 30.4 | 10.2 | 18.2 | 26.1 | 7.4 | 14.0 | 23.0 | 8.1 | 16.4 | 24.7 | 9.0 |
| Image + Text | 17.1 | 29.5 | 40.5 | 13.1 | 22.2 | 31.9 | 11.5 | 20.1 | 29.2 | 9.8 | 20.0 | 28.9 | 12.9 |
| Combiner (CC3M, Ours) | 19.7 | 31.7 | 42.1 | 16.2 | 27.3 | 37.5 | 16.6 | 27.7 | 37.2 | 18.0 | 32.2 | 41.6 | 17.6 |

method in Table 2. Furthermore, when embedding an object name, we ensemble over the 80 standard CLIP prompts from [60]. For the object-based baselines, we use a Detic model with the strongest Swin-B backbone [43], trained for open-vocabulary detection on the ‘base’ classes of the LVIS dataset [22]. For the first detection stage, we select a confidence threshold which optimizes downstream Recall@1 on the ‘Focus Object’ evaluation (a confidence of 0.2).

F. Extended Related Work

We briefly describe work from the *text-based image-editing* domain and discuss how it relates to our work. Generative image-editing [6, 49, 53] is a popular task in which, given a reference image and some condition, a sensible edit of the image is *generated*. Recently, with the advent of widely available large-scale generative models [62], there has been substantial work which considers the prompt as a text-condition, and uses CLIP text embeddings to guide the generation process [2, 5, 14, 20, 33, 38]. As such, the inputs and outputs of image-editing models are similar to those considered in this work. However, we highlight our work focuses on the *representation* and *retrieval* of existing images, rather than the synthesis of new ones.

We also note recent work which considers a task similar to ‘Change an Object’ in GeneCIS, in the context of compositional learning [48]. Similar to work detailed in § 2, [48] trains on a finite set of categories and considers only one of the four areas of the conditional similarity space which we propose here. Finally, we highlight [78], which considers contrastively training a single backbone with multiple heads, each of which is invariant to different data augmentations. This work shares similar motivations to ours – that there are many notions of image similarity – though they train with a pre-determined and fixed set of data augmentations (and hence fixed concepts of similarity).

G. Qualitative Examples

G.1. GeneCIS examples

We provide example templates from the GeneCIS benchmark tasks in Figures 7 to 10. Differently to Figure 2, we show the entire curated retrieval templates of 10-15 target images, as well as the reference image (leftmost, yellow) and condition text (blue oval). As discussed in § 4, all

gallery images are *implicitly similar* to the reference image or condition. The ‘positive’ target image is the *most similar given the condition*.

G.2. Model Predictions

We show qualitative results of our model and CLIP-only baselines in Figure 11. We show instances where our model fails in Figure 12 along with the ‘correct’ target image and the prediction of the Image-Only CLIP baseline.

G.3. Mined Triplets from CC3M

We show examples of training triplets which we *automatically* mine from CC3M [66] in Figure 13.

H. Attributions

Figure 1: Thomas Hawk, CC BY-NC 2.0 (<https://creativecommons.org/licenses/by-nc/2.0/>), via Flickr.

Figure 4, image of horse on canvas: Simon Kozhin, CC BY-SA 3.0 (<https://creativecommons.org/licenses/by-sa/3.0/>), via Wikimedia Commons.

I. Acknowledgements

We would like to thank Weidi Xie, Liliame Momeni, Mannat Singh and Kalyan Vasudev Alwala for valuable discussions on this work. Sagar is supported by a Facebook AI Research Scholarship.



Figure 7. Focus on an Attribute example templates.

matte table




"shining"




Same Object
Condition Attribute




Wrong Object
Condition Attribute





Wrong Object
Condition Attribute



Wrong Object
Condition Attribute




| | | | | | |
|---|---|---|---|--|---|
| <p>Wrong Object Condition Attribute</p>  | <p>Wrong Object Condition Attribute</p>  | <p>Wrong Object Condition Attribute</p>  | <p>Wrong Object Condition Attribute</p>  | <p>Wrong Object Condition Attribute</p>  | <p>Wrong Object Condition Attribute</p>  |
| <p>Same Object Wrong Attribute</p>  | <p>Same Object Wrong Attribute</p>  | <p>Same Object Wrong Attribute</p>  | <p>Same Object Wrong Attribute</p>  | <p>Same Object Wrong Attribute</p>  | |

short carrots




"long"



Same Object
Condition Attribute



Wrong Object
Condition Attribute



Wrong Object
Condition Attribute



Wrong Object
Condition Attribute





| | | | | | |
|---|---|---|---|--|---|
| <p>Wrong Object Condition Attribute</p>  | <p>Wrong Object Condition Attribute</p>  | <p>Wrong Object Condition Attribute</p>  | <p>Wrong Object Condition Attribute</p>  | <p>Wrong Object Condition Attribute</p>  | <p>Wrong Object Condition Attribute</p>  |
| <p>Same Object Wrong Attribute</p>  | <p>Same Object Wrong Attribute</p>  | <p>Same Object Wrong Attribute</p>  | <p>Same Object Wrong Attribute</p>  | <p>Same Object Wrong Attribute</p>  | |

Figure 8. Change an Attribute example templates.



Figure 9. Focus on an Object example templates.



Figure 10. Change an Object example templates.



Figure 11. Qualitative results from our method showing instances where our model predicts correctly.

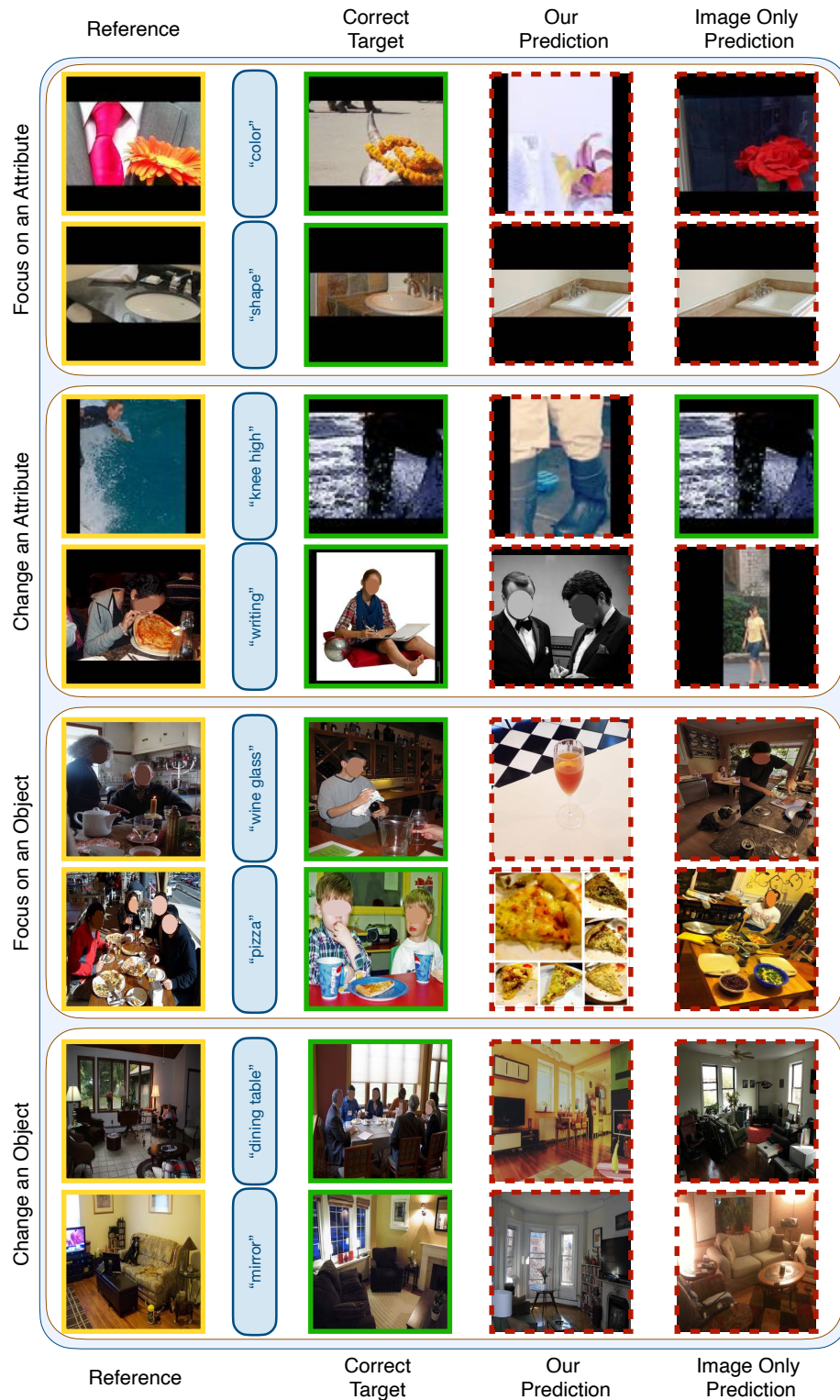


Figure 12. Qualitative results from our method showing instances where our model *fails*.

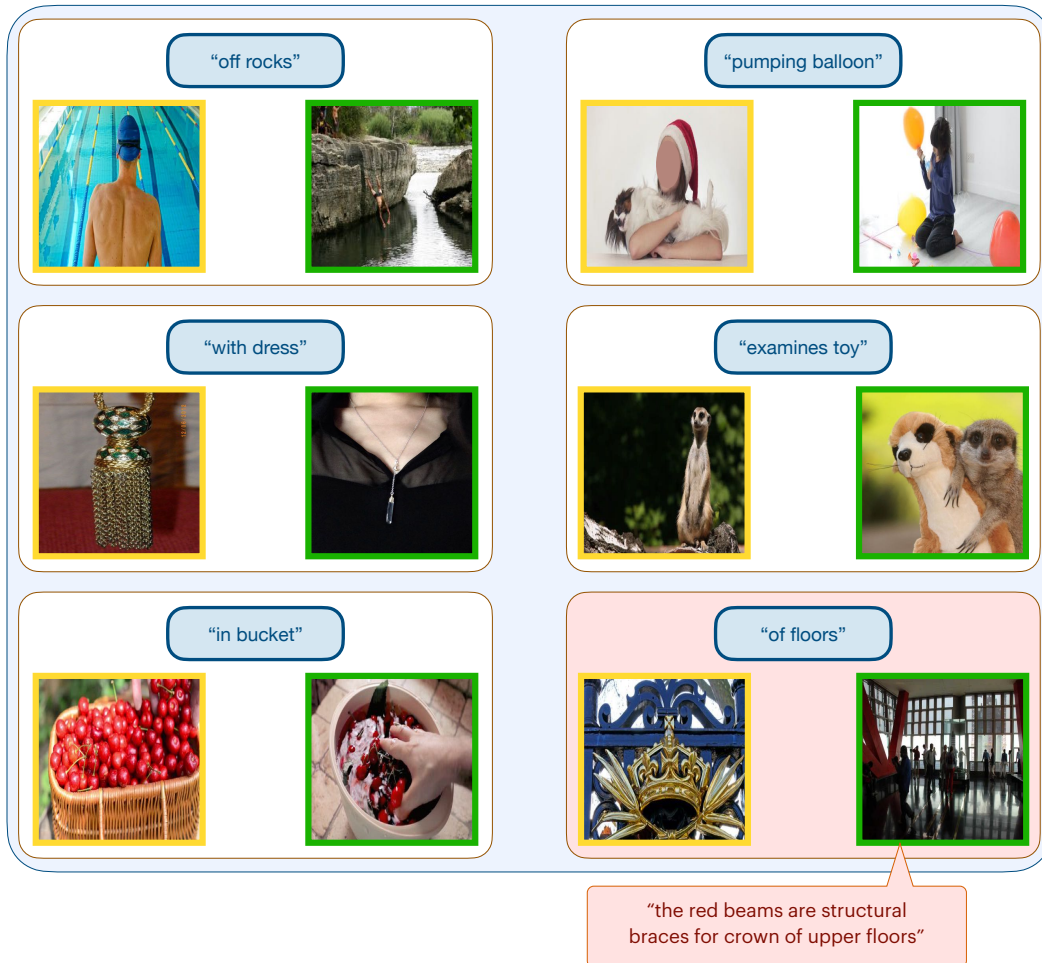


Figure 13. Examples of mined triplets from CC3M [66] as described in § 5.2. In each triplet, the text condition (blue oval) links the reference image (left) to the target image (right). We show an instance where a noisy triplet is produced in the bottom right. The caption (shown in a speech bubble) incurs a misleading parsed relationship of ‘Crown’ → ‘of’ → ‘floors’.