Is CLIP ideal? No. Can we fix it? Yes!
–Supplementary Material–

A Extended Related Work **12**

B Experimental Details **13**
B.1. Implementation Details 13
B.2. Datasets 13
B.3. Pseudocode 14

C Ablation Studies **14**
C.1. Impact of DCSMs 14
C.2. Classification Performance 15
C.3. Scaling Analysis 15

D Empirical Observations of CLIP shortcomings **15**

E Definitions and Proofs **15**
E.1. Definitions Addendum 15
E.2. Conditions Addendum 16
E.3. Visualization of proofs 16
E.4. Lemma 2. Addendum 16
E.5. Contradiction for Condition 1 and 3 17
E.6. Contradiction for Condition 1 and 4 18

F. Open Vocabulary Experimental Details **19**
F.1. LLM System Prompts 20

A. Extended Related Work

Vision Language Models. Recent advancements in VLMs have significantly bridged the gap between vision and language. A seminal work in this area is CLIP [49], which demonstrated that large-scale contrastive learning can effectively align image and text representations into a shared embedding space. Building upon its core principles of large-scale contrastive pretraining and joint representation learning, several subsequent works have explored alternative architectures and training paradigms, including ALIGN [17], FILIP [67], SLIP [38], ALBEF [25], and CoCa [68], to name a few. Autoregressive VLMs [33, 34] have emerged as compelling alternatives to CLIP by jointly attending to both text and image embeddings. Neurosymbolic program synthesis methods like ViperGPT [57], VisProg [15], and VADAR [36] also mitigate some of CLIP's limitations by formulating the text-image semantic distance acquisition as several subproblems. While these methods are more comprehensive than CLIP and can specialize in complex visual reasoning, they are orders of magnitude more expensive to infer and do not offer the same simplicity or gradient retention as CLIP, limiting their downstream applicability. In fact, most of these models rely on a CLIP-like model's latent space as a submodule. As such there remains a strong motivation to continue refining and extending CLIP-like architectures by addressing their inherent shortcomings.

Note that at a very coarse level, DCSMs are analogous to the QK^T matrix computed between the text token embeddings and image patch embeddings during the attention loop in autoregressive VLMs. However, QK^T matrices are dynamically computed during attention using learned projections, while DCSMs are static post-encoder similarity measures. These VLMs are also trained on large data corpuses for use cases beyond simple text to image distance matching. Often, while the initial image encoder remains frozen, the training pipeline includes learning the optimal initial projection for text token embeddings in addition to the downstream LLM decoder. In this work, we pursue the more narrow aim of maximizing the potential for CLIP to produce correct semantic image-text scores while keeping both text and image embeddings frozen.

Empirical Limitations of CLIP. A growing body of work has revealed several limitations of CLIP in handling complex visual-text interactions. One major issue is its difficulty in distinguishing between different attribute-concept bindings in multi-object scenes [8, 23, 39]. For example, the text prompt "A purple sphere" will have very high cosine similarity with an image that contains a purple cube and a yellow sphere, despite the yellow attribute not belonging to the spherical object. Lewis *et al.* propose CLEVR-bind as a simple benchmark which isolates the attribute binding capability of VLMs. More comprehensive natural-language benchmarks targeting attribute binding include Attribute, Relation, and Order (ARO), Sugarcrepe [16], VL-checklist [75], and Multimodal Visual Patterns (MMVP) [58]. Additional studies have noted that CLIP's text embeddings often behave like "bag-of-words" in practice, leading to imsinterpretations of object layouts or conflates multiple entities within a single scene [23, 39, 69]. Yuksekgonul *et. al* specifically propose WhatsUP as a benchmark which isolates spatial reasoning capacities of VLMs. In addition, the suite of compositional understanding benchmarks (ARO, VL-checklist, Sugarcrepe, MMVP) also include captions that require spatial reasoning. Another notable failure mode is CLIP's inability to accurately represent negation [1, 55]. In response, Alhamoud *et al.* develop NegBench to specifically assess how well VLMs handle various forms of negatory sentences. The aforementioned compositional benchmarks further challenge models with captions that require proper negation understanding.

Some other criticisms of CLIP are its inability to generalize to different reference frames [74] or to count [42]. While these issues represent additional challenges for CLIP, they are beyond the scope of our current work.

Proposed Solutions to CLIP Limitations. The most prominent method of corrections have been to change the training data distribution, such as retraining or fine-tuning CLIP with hard negative or positive examples [1, 21, 42, 48, 55, 62, 69]

or increasing the training data for more comprehensive and longer captions [30, 63]. However, simply scaling the training data to mitigate specific problems will often lead to reduced generalization [16, 35, 71].

Some engineering solutions include using object detectors to segment images into smaller ROIs [27], adding attribution tracing for correct text-image pairs during training [28], patch clustering for semantic segmentation [53], using chain of thought spatial reasoning [9] or altering the self attention mechanism in the vision encoder [61]. Simpler solutions may be querying CLIP with multiple descriptions [37]. All of these methods require retraining CLIP from scratch, or typically adding a heavy-handed component to single out the objects in a scene.

B. Experimental Details

B.1. Implementation Details

By convention the rows of the dense map correspond to one text token embedding, and the columns to one image patch. Every text and image pair creates a DCSM of shape (30,197), where 30 is the maximum number of text tokens and 197 is the number of 16x16 image patches in an image of shape 224x224. Text prompts shorter than 30 tokens are padded with EOS tokens. FRs are therefore of the shape (num-image-patches, 1). For example, for the sentence "An image of a circle above a triangle", the word *above* is a functional word and the corresponding row in the DCSM gets replaced with the respective FR in the lookup table. For synonymous functional words, we use a single FR. (Somewhat unusually, we consider "front","below", and "behind","above" to be synonymous, as DCSMs use a 2D fixed frame of reference due to the topology being represented by the patch index.) The DCSMs are z-score normalized for stable training.

For all training and experiments, we use a lightweight CNN with 2 convolutional layers and a hidden dimension of 128. In addition to a 20-fold reduction in parameters, we train our network with a batch size of 8, a 4000-fold decrease from the original mini-batch size of 32,768. In fact, the sum of all our training data per model is smaller than this number.

Our model outputs a single score for each image and text pair DCSM. During training, we use a contrastive cross-entropy loss as with the original CLIP [50]. We train with the Adam optimizer with learning rate initialized at $1e-3$. One model is trained with a curated synthetic dataset and another with COCO 2017 training split. Dataset curation is detailed below.

B.2. Datasets

We train our pipeline on two different datasets - one synthetic dataset composed of open-source 3D assets from Objaverse [12] placed upon randomized backgrounds, and another generated from COCO-train-2017 [32].

We are mainly interested in labeling images with text prompts that lie in the Condition 2,3,4 category. That is, for both the synthetic case and the COCO-train case, we generate a dataset for attribute binding, spatial relationships/localization, and negation. Samples from each dataset are shown in Fig. 6.

Attribute Binding Dataset Every image in this dataset includes two distinct objects of unique colors or sizes. For each image, we generate a "hard negative". So if there is an image with a "red cow and purple ghost", we also generate an image with "red ghost and purple cow". This means that every sample has a positive and negative image, and two positive and negative captions each. The positive image contains object A with attribute A_{att}, and object B with attribute B_{att}. The negative image contains the same objects but with swapped attributes. The positive caption options are: (P1) "A_{att} A and B_{att} B" and (P2) "B_{att} B and A_{att} A". The negative caption options are: (N1) "A_{att} B and B_{att} A" and (N2) "B_{att} A and A_{att} B". We generate 5,402 images for this dataset, which makes 2701 samples with associated opposites.

For the COCO-train set, we use a natural language processing library to extract adjective-noun pairs in the natural language captions, and select images that have at least two distinct objects A, B with distinct attributes A_{att}, B_{att}. Captions follow the same format as above. We select 8,547 images from COCO-train towards this dataset.

Spatial Relationships and Localization Similarly as above, we generate synthetic images where one object is placed either above, below, to the left, or to the right of, another object with random jitter. For every positive image where A is *rel* to B, there is a negative image where A is rel_{opp} to A. Here, (above, below) are opposite relation pairs, as are (left of, right of). The positive caption options are: (P1) "A *rel* B" and (P2) "B rel_{opp} A". The negative caption options are: (N1) "A rel_{opp} B" and (N2) "B *rel* A". We generate 11,324 images for this dataset, which makes 5662 samples with associated opposites.

For the COCO images, we choose images where at least two distinct objects are present, and use the relationship between their bounding boxes to validate that they satisfy the definition of one of the spatial relationships being considered. Note that, as we are using patch location on the image to preserve topology, we use a fixed frame of reference to determine the meaning of "above", "below", "left", and "right". To make a negative version of the image, we use the CutMix technique [70] to swap the image content in the two bounding boxes. With the positive images from COCO-train and generated hard negatives, there are 11,502 images in this dataset, which makes 5751 samples with associated opposites.

Negaton Generating negatory captions is tricky. With contrastive training, an entirely negatory caption (e.g., "An

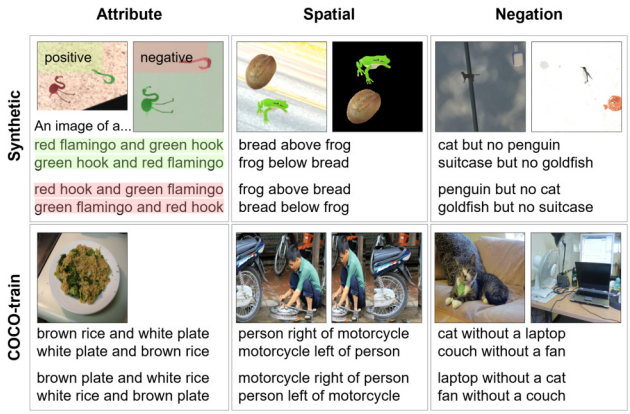

Figure 6. Overview of Dataset Curation. Each box is a dataset. The left image is a positive pair with the top two captions, and the right image is a positive pair with the bottom two captions. The COCO-train Attribute set does not have a hard negative image counterpart.

image without a turtle") necessitates that all other images in the batch be a positive sample (e.g., they must now all contain a turtle).

As such, for the first pass for the models trained with synthetic data, we generate images with two random objects $A1$ and $A2$, and choose as its hard negative another image which contains two non-overlapping objects, $B1$ and $B2$. For the negatory term $\neg$, we choose between ['but not', 'and no', 'without']. The positive caption options are: (P1) "$A1 \neg B1$" and (P2) "$A2 \neg B2$". The negative caption options are: (N1) "$B1 \neg A1$" and (N2) "$B2 \neg A2$". We generate 10,000 images for this dataset.

For COCO-train we select two images that have two distinct object labels, and generate captions the same way. We select 10,000 images toward this dataset.

B.3. Pseudocode

Below we illustrate the process for extracting DCSMs.

```
dense_image_features = image_encoder(I).
    last_hidden_state.unsqueeze(1)
# Shape: (batch_size, 1, iseq, embed_dim)

dense_text_features = text_encoder(T).
    last_hidden_state.unsqueeze(0)
# Shape: (1, batch_size, tseq, embed_dim)

dcsm = einsum( "bqie, lpte -> bpit",
    dense_image_features, dense_text_features
    )
# Shape: (batch_size, batch_size, iseq, tseq)

dcsm = add_functional_rows(dcsm, lookup_list)

out = lightweight_cnn(dcsm)
# Shape: batch_size, batch_size

labels = eye(out.shape[0])
# Shape: batch_size, batch_size

loss = CE(out, labels) + CE(out.t(), labels.t
    ())
```

Figure 7. Pytorch-like code for training our model with DCSMs

C. Ablation Studies

C.1. Impact of DCSMs

Model	WhatsUP	COCO_{2obj}
Ours - DCSM (CNN)	**62.6**	**70.9**
Ours - DCSM (CNN)$_{w/oFR}$	48.6	55.5
Ours - DCSM (ViT)	29.4	47.0
CLIP - ViTB/16	30.5	45.9
CLIP - ViTB/16$_{f.t.\ synth}$	25.5	49.1
CLIP - ViTB/16$_{MLP\ scorer}$	25.4	53.6

Table 4. The DCSM networks were trained with synthetic data.

In Sec 5. we said that it follows from our analysis that neither a fine-tuned/reprojected CLIP embedding space, nor a learned scoring module, could alone be the fix to CLIP's fundamental shortcomings. To verify this conclusion, we perform a series of ablations. First, we finetune OpenAI's CLIP (ViTB/16) with the same synthetic dataset used for our model training. We also train a MLP scoring module which takes concatenated text and image embeddings as input to output a score, again on the same synthetic dataset. Both attempts fail to improve performance on WhatsUP and COCO-spatial, resulting in accuracies near chance (25% and 50%, respectively).

Further, we alter our training pipeline in two different ways to assess the need for a CNN as well as the FRs. Removal of the FRs decreases performance overall, but the increased information capacity from using the DCSMs and the downstream network still allows the model to perform above fine-tuned CLIP models. Replacement of the CNN with a comparably small ViT, with a patch size of 2 and 2 layers of 4 attention heads, resulted in another near-chance performance on the datasets. The ViT appears more prone to overfitting to the training set, as it does not have the imposed constraint of pattern-recognizing kernels as in CNNs.

All networks were trained with learning rate 1e-3 with Adam for 27 epochs. Under minimal compute, the CNN generalized much better. Fine-tuning CLIP projection layers with the same dataset for the same number of epochs did not result in any noticeable performance increase.

C.2. Classification Performance

A known problem with VLMs using CLIP embeddings is the decline in classification capacity [73]. We evaluate our model on two classification datasets and find that, despite being trained with simple prompts and no image classification captions, the reduction in classification performance is minimal compared to those observed in BLIP or LLaVA.

Model	Caltech101	Flowers102
CLIP-ViT B/16 [49]	**82.6**	**67.7**
Ours - DCSM + synth	**77.8**	**52.9**
Ours - DCSM + coco	**79.2**	**43.7**
BLIP2-2.7B [26]	22.3	14.2
IBLIP-7B [11]	58.4	26.8
LLaVA1.5-7B [34]	62.1	10.2

Table 5. Classification Accuracy of VLMs. Top three scores per dataset are bolded.

C.3. Scaling Analysis

In this work, we showcase a very small and computationally light network and training pipeline for the DCSM method. To verify that this method will scale with increasing data, we perform a scaling analysis. Fig. 8 shows the results. The x axis is the approximate number of samples from the curated COCO2017 training set. From this we see that our training pipeline is likely to scale with increasing the dataset.

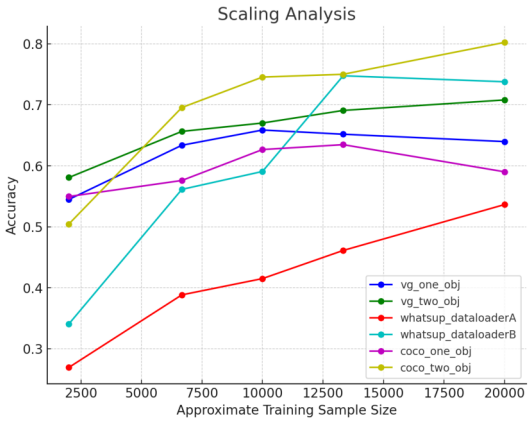

Figure 8. Result of linearly scaling training data. x values are approximate dataset sizes.

D. Empirical Observations of CLIP shortcomings

In Fig. 9 we show the empirical effects of the superposition derived in Lemma 1. In summary, the figure serves to show how an image with an increasingly greater number of objects present embeds increasingly farther from the text label for any one of those objects in the CLIP latent space. The degree of this effect is such that beyond 6-8 objects in one image, CLIP embeddings of random noise images are similarly close to those object text labels.

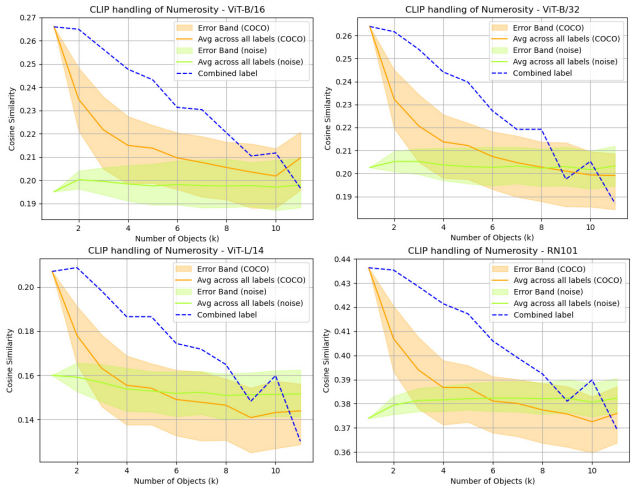

Figure 9. For each image in COCO-validation, we identify k objects with labels. Then we take the cosine similarity between all k labels and the image. The orange line shows the average cosine similarity for images with k objects and all labels that appear in the image. The blue dotted line shows the cosine similarity for a label which combines all k labels and the image being considered, averaged for each k. The green plot shows the average cosine similarity score between 5 random noise images and the labels of all COCO-validation images. The error bands indicate the 25th and 75th quartile.

E. Definitions and Proofs

E.1. Definitions Addendum

Definition 1. For x = the concept of a bird, its mapping in $\mathbb{I}$ is an image of a bird, and its mapping in $\mathbb{T}$ is “bird". For all concepts x in $\mathbb{V}$, there is an equivalence class in $\mathbb{I}$ and $\mathbb{T}$ that unambiguously represent that concept, respectively. We narrow the scope of all images to 1 representative image per concept. So for the object concept of a “bird", there is one image which is equivalent. Similarly, for each concept x there is only 1 corresponding $d \in \mathbb{T}$.

We also choose M object concept elements toward the subset $\mathbb{V}$. In the manner of large ontologies (such as ImageNet1000), we take it to be true that there exist at least $M \leq 1000$ object classes which are mutually exclusive. This means that for any representative image in the set of M, an expert human reviewer would be able to assign it to that unique object class. These object classes must be *subordinate categories* on any hierarchical directed tree of visual concepts. (An example concept hierarchy: *car* and *bus* are

lower-level concepts that are below *vehicle*.) This ensures no two concepts in $\mathbb{V}$ overlap in semantics.

E.2. Conditions Addendum

Condition 1. Semantic separability means: if human annotators would agree that a text caption appropriately describes an image, the embeddings for that caption and image should have high cosine similarity. Conversely, the score should be low if an annotator deems the caption to be inappropriately matched to the image.

Notably, this condition does not require correct hierarchical organization among concepts or semantically accurate placement of synonymous phrases. In reality there exist more than N distinct concept categories and each category may have synonymous text representations and multiple instances and viewpoints of the representative visual object. Our goal is to set up extremely minimal geometric requirements for C to evaluate whether they are possible to attain.

This condition must be satisfied for C to perform zero-shot image classification, retrieval, and semantic similarity search. Example benchmarks that pose this challenge include Imagenet, COCO, LAION, and many more.

Condition 2. This condition must be satisfied for C to perform tasks where the model must identify attributes associated with different objects in a scene. This could be towards scene understanding, vision question answering, or accurate image retrieval. Specific datasets include CLEVR-bind, NCDataset-grayscale, VL-checklist, Sugarcrepe, ARO, and MMVP.

Condition 3. This condition must be satisfied for C to to perform tasks that require compositional image understanding. This could be for image captioning, text-guided image generation, spatial navigation, and more. Specific datasets include WhatsUP, Coco-spatial, MMVP, etc.

Condition 4. This condition must be satisfied for C to perform well on vision-language tasks that include prompts with negations. Note that this condition is a very relaxed interpretation of negation: Strictly semantically speaking,"not X" is a correct semantic pair with any image that does not have X, requiring a cosine similarity near 1. But the definition of the negation condition we impose does not require such granularity. Again, we seek to pose minimal constraints to identify whether there is some version of negation CLIP could attain under ideal settings. Specific datasets that require negation understanding include NegBench, as well as other compositional benchmarks like VLM-checklist, Sugarcrepe, or MMVP.

E.3. Visualization of proofs

Fig. 10 illustrates the proofs visually.

E.4. Lemma 2. Addendum

Derivation of p. We define the perturbed vector:

$$\mathbf{i}(x_a) = (1-\delta)\,\mathbf{i}(x) + p\mathbf{t}(a), \|\mathbf{i}(x_a)\|^2 = 1 \tag{10}$$

Expanding the norm, we have:

$$\begin{aligned} \|\mathbf{i}(x_a)\|^2 &= \|(1-\delta)\mathbf{i}(x) + p\mathbf{t}(a)\|^2 \\ \|\mathbf{i}(x_a)\|^2 &= (1-\delta)^2\|\mathbf{i}(x)\|^2 + p^2\|\mathbf{t}(a)\|^2 \\ &\quad + 2(1-\delta)p\mathbf{i}(x)\cdot\mathbf{t}(a) \end{aligned} \tag{11}$$

Since $\mathbf{i}(x)$ and $\mathbf{t}(a)$ are unit vectors, this simplifies to:

$$\|\mathbf{i}(x_a)\|^2 = (1-\delta)^2 + p^2 + 2(1-\delta)p\cos\theta \tag{12}$$

where $\cos\theta = \mathbf{i}(x)\cdot\mathbf{t}(a)$. For $\mathbf{i}(x_a)$ to be a unit vector, we set the right hand side to 1:

$$\begin{aligned} (1-\delta)^2 + p^2 + 2(1-\delta)p\cos\theta &= 1 \\ -2\delta + \delta^2 + p^2 + 2(1-\delta)p\cos\theta &= 0 \end{aligned} \tag{13}$$

Notice that this is now a quadratic equation in p:

$$p^2 + 2(1-\delta)p\cos\theta - (2\delta - \delta^2) = 0. \tag{14}$$

Use the quadratic formula:

$$p = \frac{-2(1-\delta)\cos\theta \pm \sqrt{[2(1-\delta)\cos\theta]^2 + 4(2\delta-\delta^2)}}{2} \tag{15}$$

We can simplify the above to find the correct value of p that ensures $\mathbf{i}(x_a)$ is a unit vector.

$$p = -(1-\delta)\cos\theta \pm \frac{\sqrt{4(1-\delta)^2\cos^2\theta + 8\delta - 4\delta^2}}{2}. \tag{16}$$

Analysis for noise vectors. In Lemma 2 we derived

$$\boxed{\begin{aligned} \mathbf{i}(x_a, y_b) &= \frac{(1-\delta)(\mathbf{i}(x)+\mathbf{i}(y)) + p\mathbf{t}(a) + q\mathbf{t}(b)}{2} \\ &= \mathbf{i}(x_b, y_a) \end{aligned}} \tag{17}$$

using $\mathbf{i}(x_a) = (1-\delta)\mathbf{i}(x) + p\mathbf{t}(a)$.

Now we show that analytically, the inclusion of a noise vector ε does not change the results. Specifically, we want to simulate

$$\mathbf{i}(x_a) = \frac{(1-\delta)\mathbf{i}(x) + \delta\mathbf{t}(a) + \varepsilon}{\|norm\|}$$

for some randomly sampled ε and δ.

In Fig. 11 we showcase the results of sampling ε from a standard normal distribution, with varying weights.

We observe the following relations remain consistent:

- As expected of randomly initialized vectors in high dimensions: $\mathbf{i}(x)\cdot\mathbf{t}(a) \approx 0$
- For some unrelated object-attribute pairs, their image embeddings are roughly orthogonal as well: $\mathbf{i}(w_c, z_d)\cdot\mathbf{i}(x_a, y_b) \approx 0$
- The strong conclusion from Lemma 2 is approximately always true: $\mathbf{i}(x_b, y_a)\cdot\mathbf{i}(x_a, y_b) \approx 1$

regardless of $\mathbf{i}(x_a)\cdot\mathbf{t}(a), \mathbf{i}(x)\cdot\mathbf{i}(x_a)$. As such, we see that even if there is noise in the superpositions described in Lemma 1 and 2, C still cannot disambiguate between different pairings of the same two attributes and two objects.

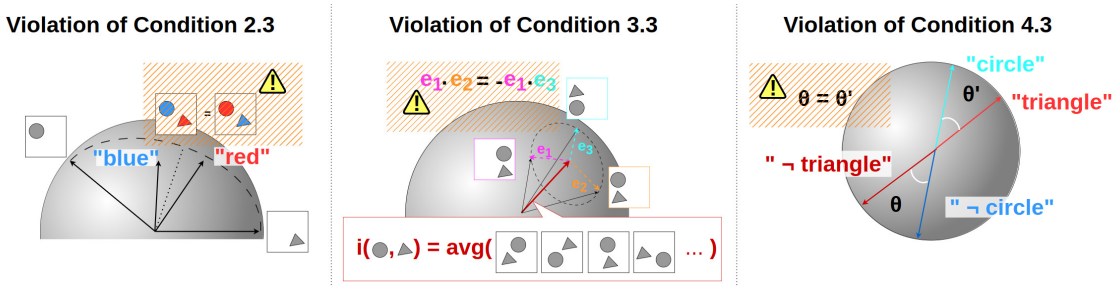

Figure 10. Conceptual Overview of Contradictions

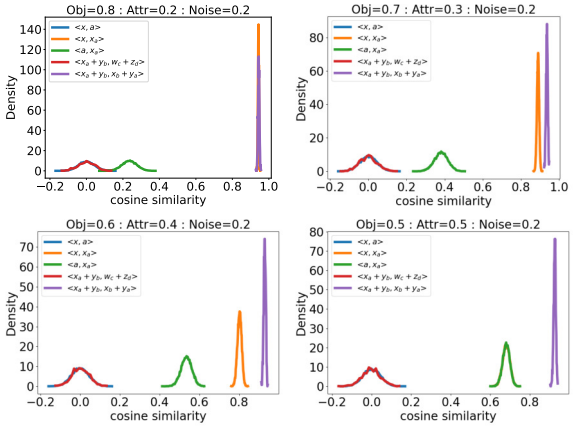

Figure 11. Titles of each subplot indicate the weights of the different components composing $\mathbf{i}(x_a, y_b), \mathbf{i}(x_b, y_a)$. Obj= indicates weight of the object concept embeddings $\mathbf{i}(x), \mathbf{i}(y)$, Attr= indicates weight of the attribute text embeddings $\mathbf{t}(a), \mathbf{t}(b)$, and Noise= indicates weight of the noise vector ε. In the legend, object concepts and attribute embeddings are denoted in shorthand: $x = \mathbf{i}(x), a = \mathbf{t}(a)$, and so on.

E.5. Contradiction for Condition 1 and 3

Now we show Condition 3 cannot be met if Condition 1 is met. Below we will show two impossibility cases and prove them.

Lemma 3. *C cannot accurately represent both the distance between spatial locations and relationships at the same time.*

Proof. We first derive $\mathbf{i}(x, g^{<rel>}, y)$ for two antonymous $g^{<rel>}$ to satisfy Condition 1.2. Then we consider an example scenario with two objects, two spatial relationships, and two localizing terms. We will find that for three sample images, the cosine similarities between their embeddings and four textual clauses will have to contradict Condition 3 for some pairs.

For two concepts x and y, their combined embedding is Eq. (3). Now, if we want to express some compositional relationship $g^{<rel>} \in \mathbb{G}$ between x and y such that $\mathbf{i}(x, g_1^{<rel>}, y) \neq \mathbf{i}(x, g_2^{<rel>}, y)$, we can write

$$\begin{aligned} \mathbf{i}(x, g_1^{<rel>}, y) &= (1-\delta)\,\mathbf{i}(x, y) + \mathbf{v}_1 \\ \mathbf{i}(x, g_2^{<rel>}, y) &= (1-\delta)\,\mathbf{i}(x, y) + \mathbf{v}_2 \end{aligned} \tag{18}$$

where $\delta \ll 1$ and $\mathbf{v}_1 \neq \mathbf{v}_2$. Similar to Lemma 2, $\mathbf{v}$ is a small location-specific component, as $\mathbf{i}(x, g^{<rel>}, y)$ must remain close to $\mathbf{i}(x, y)$ per Condition 1.2.

Let $g_L^{<rel>} = g_L$ be the relational concept whose equivalent mapping in $\mathbb{T}$ is "_ left of _", and $g_R^{<rel>} = g_R$ "_ right of _". Then we can write:

$$\begin{aligned} \mathbf{i}(x, g_L, y) &= (1-\delta)\,\mathbf{i}(x, y) + \mathbf{e}_{\perp,L} \\ \mathbf{i}(x, g_R, y) &= (1-\delta)\,\mathbf{i}(x, y) + \mathbf{e}_{\perp,R} \end{aligned}$$

where $\mathbf{e}_{\perp,L}, \mathbf{e}_{\perp,R}$ both lie in the orthogonal error subspace (of dimension $N-1$) and have fixed magnitude $\sqrt{2\delta - \delta^2}$ such that $\mathbf{i}(x, g_L, y)$ is a unit vector. Notice that we cannot use the intuition from Lemma 2 that v must be composed of the textual component of g_L - an image with a melon above a bed does not intuitively need to embed closely with the text embedding for "above".

Then to *maximize* the Euclidean distance $\|\mathbf{i}(x, g_L, y) - \mathbf{i}(x, g_R, y)\|$, we must choose $\mathbf{e}_{\perp,R} = -\mathbf{e}_{\perp,L}$. Since $(x, g_L, y), (x, g_R, y) \in \mathbb{S}$, they also have an equivalent item in $\mathbb{T}$.

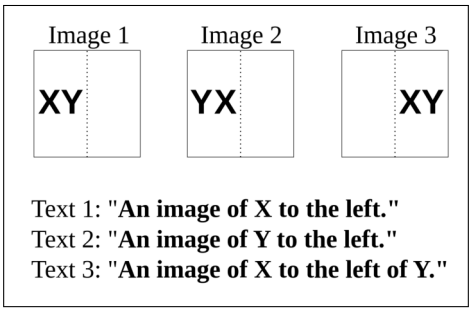

Figure 12. Simple setup. Here the only relations that exist are "_ left _", "_ right of _", "_ to the left", and "_ to the right".

Now we formulate a proof by contradiction. Consider three images and three text prompts, as shown in Fig. 12. In addition to g_L, g_R, we also introduce $g_l^{<loc>} = g_l, g_r^{<loc>} = g_r$, to represent "_ to the left" and "_ to the right" in $\mathbb{T}$,

respectively. These are the only four compositional concepts we consider for simplicity, but conclusions will generalize.

We denote $(1-\delta)\mathbf{i}(x,y) = \mathbf{e}_{||}$, the three images as $\mathbf{i}(\text{image}_k) = \mathbf{i}_k$ and $\mathbf{t}(\text{text}_j) = \mathbf{t}_j$, for $k, j \in 1,2,3$. $\mathbf{i}_k = \mathbf{e}_{||} + \mathbf{e}_{\perp,k}$ where $\mathbf{e}_{||}$ is the shared parallel component for all $\mathbf{i}_k$.

Consider some images $\mathbf{i}_w, w > 3$, where x, y do not have g_L, g_R, g_l or $, g_r$ as a location or relationship. For example $\mathbf{i}_4$ could be an image of x on top of y in the center of the frame. In this case,

$$\mathbf{i}_k \cdot \mathbf{i}_w = (\mathbf{e}_{||} + \mathbf{e}_{\perp,k}) \cdot (\mathbf{e}_{||} + \mathbf{e}_{\perp,w}) = |\mathbf{e}_{||}|, k \in (1,2,3)$$

as these image pairs share no compositional aspects in common. Since $\mathbf{i}_1, \mathbf{i}_2$ share the same localization for the objects present, satisfaction of Condition 3.2 requires that $\mathbf{i}_1 \cdot \mathbf{i}_2 > |\mathbf{e}_{||}|$. Similarly, as $\mathbf{i}_1, \mathbf{i}_2$ share the same relationship between the objects present, satisfaction of Condition 3.3 requires that $\mathbf{i}_1 \cdot \mathbf{i}_3 > |\mathbf{e}_{||}|$.

This could only happen if embeddings of images 1,2 and i,3 have perpendicular components that are partially parallel. In other words, **in order to satisfy Conditions 3.2 and 3.3, there must exist a C in which $\mathbf{e}_{\perp,1} \cdot \mathbf{e}_{\perp,2} > 0$ and $\mathbf{e}_{\perp,1} \cdot \mathbf{e}_{\perp,3} > 0$.**

For each image in Fig. 12, the embeddings must optimize the following local similarities:

$$\begin{aligned} \mathbf{i}_1 &= \underset{\mathbf{i}_1}{\text{argmax}}(\mathbf{i}_1 \cdot \mathbf{t}_1 + \mathbf{i}_1 \cdot \mathbf{t}_2 + \mathbf{i}_1 \cdot \mathbf{t}_3) \\ \mathbf{i}_2 &= \underset{\mathbf{i}_2}{\text{argmax}}(\mathbf{i}_2 \cdot \mathbf{t}_1 + \mathbf{i}_2 \cdot \mathbf{t}_2 + \mathbf{i}_2 \cdot -\mathbf{t}_3) \\ \mathbf{i}_3 &= \underset{\mathbf{i}_3}{\text{argmax}}(\mathbf{i}_3 \cdot -\mathbf{t}_1 + \mathbf{i}_3 \cdot -\mathbf{t}_2 + \mathbf{i}_3 \cdot \mathbf{t}_3) \end{aligned} \tag{19}$$

subject to $\|\mathbf{i}_k\| = 1$. This allows us to solve for $\mathbf{e}_{\perp,k}$s. For $k = 1$:

$$\begin{aligned} \mathbf{e}_{\perp,1} = \underset{\mathbf{e}_{\perp,1}}{\text{argmax}}((\mathbf{e}_{||} + \mathbf{e}_{\perp,1}) \cdot \mathbf{t}_1 + ((\mathbf{e}_{||} + \mathbf{e}_{\perp,1}) \cdot \mathbf{t}_2 + \\ ((\mathbf{e}_{||} + \mathbf{e}_{\perp,1}) \cdot \mathbf{t}_3) \end{aligned} \tag{20}$$

subject to $\left\|\mathbf{e}_{\perp,1}\right\| = \sqrt{2\delta - \delta^2}$. $\mathbf{e}_{||}$ is fixed so this becomes:

$$\begin{aligned} \mathbf{e}_{\perp,1} &= \underset{\mathbf{e}_{\perp,1}}{\text{argmax}}(\mathbf{e}_{||} \cdot (\mathbf{t}_1 + \mathbf{t}_2 + \mathbf{t}_3) + \mathbf{e}_{\perp,1} \cdot (\mathbf{t}_1 + \mathbf{t}_2 + \mathbf{t}_3)) \\ &= (\mathbf{t}_1 + \mathbf{t}_2 + \mathbf{t}_3) \cdot \sqrt{2\delta - \delta^2} \end{aligned} \tag{21}$$

Similarly, we get:

$$\begin{aligned} \mathbf{e}_{\perp,2} &= (\mathbf{t}_1 + \mathbf{t}_2 - \mathbf{t}_3) \cdot \sqrt{2\delta - \delta^2} \\ \mathbf{e}_{\perp,3} &= (-\mathbf{t}_1 - \mathbf{t}_2 + \mathbf{t}_3) \cdot \sqrt{2\delta - \delta^2} \end{aligned} \tag{22}$$

Now taking the dot products, we have:

$$\begin{aligned} \mathbf{e}_{\perp,1} \cdot \mathbf{e}_{\perp,2} &= (|\mathbf{t}_1| + |\mathbf{t}_2| - |\mathbf{t}_3|) \cdot 2\delta = 2\delta \\ \mathbf{e}_{\perp,1} \cdot \mathbf{e}_{\perp,3} &= (-|\mathbf{t}_1| - |\mathbf{t}_2| + |\mathbf{t}_3|) \cdot 2\delta = -2\delta \\ \mathbf{e}_{\perp,2} \cdot \mathbf{e}_{\perp,3} &= (-|\mathbf{t}_1| - |\mathbf{t}_2| - |\mathbf{t}_3|) \cdot 2\delta = -6\delta \end{aligned} \tag{23}$$

Practically speaking, it is possible by adding more representative samples in the training dataset to change the weights of $\mathbf{t}_j$s. That is, for some $\beta_1 + \beta_2 + \beta_3 = 3$, Eqs. (21,22) could be reformulated as:

$$\begin{aligned} \mathbf{e}_{\perp,1} &= (\beta_1 \mathbf{t}_1 + \beta_2 \mathbf{t}_2 + \beta_3 \mathbf{t}_3) \cdot \sqrt{2\delta - \delta^2} \\ \mathbf{e}_{\perp,2} &= (\beta_1 \mathbf{t}_1 + \beta_2 \mathbf{t}_2 - \beta_3 \mathbf{t}_3) \cdot \sqrt{2\delta - \delta^2} \\ \mathbf{e}_{\perp,3} &= (-\beta_1 \mathbf{t}_1 - \beta_2 \mathbf{t}_2 + \beta_3 \mathbf{t}_3) \cdot \sqrt{2\delta - \delta^2} \end{aligned} \tag{24}$$

The dot products then become:

$$\boxed{\begin{aligned} \mathbf{e}_{\perp,1} \cdot \mathbf{e}_{\perp,2} &= 2\delta\left(\beta_1^2 + \beta_2^2 - \beta_3^2\right) \\ \mathbf{e}_{\perp,1} \cdot \mathbf{e}_{\perp,3} &= 2\delta\left(-\beta_1^2 - \beta^2 + \beta_3^2\right) \\ \mathbf{e}_{\perp,2} \cdot \mathbf{e}_{\perp,3} &= -2\delta\left(\beta_1^2 + \beta_2^2 + \beta_3^2\right) \end{aligned}} \tag{25}$$

Note that regardless of the reweighting, $\mathbf{e}_{\perp,1} \cdot \mathbf{e}_{\perp,2} = -\mathbf{e}_{\perp,1} \cdot \mathbf{e}_{\perp,3}$. This directly negates our previous observation that an ideal C must satisfy $\mathbf{e}_{\perp,1} \cdot \mathbf{e}_{\perp,2} > 0$ and $\mathbf{e}_{\perp,1} \cdot \mathbf{e}_{\perp,3} > 0$. As such, there exists no C which sufficiently represents both relational and objective space in the image embeddings.

□

Lemma 4. ***C* cannot accurately represent compositional concepts of different hierarchy.**

Proof. Here we will show that general prepositions are erroneously closer to all unrelated prepositions.

Some prepositions are more general than others. For example, $g_B^{<rel>} = g_B$ where $f_{G,T}(g_B) =$ "_ beside _" semantically includes both g_L and g_R. The ideal placement for $\mathbf{t}(x, g_B, y)$ should locally optimize for the following similarities:

$$\begin{aligned} \mathbf{t}(x, g_B, y) = \underset{\mathbf{t}(x,g_B,y)}{\text{argmax}} \Big[&\mathbf{t}(x, g_L, y) \cdot \mathbf{t}(x, g_B, y) \\ &+ \mathbf{t}(x, g_R, y) \cdot \mathbf{t}(x, g_B, y) \\ &- \sum_{z=1}^{|\mathbb{G} \setminus \{g_L, g_R, g_B\}|} \mathbf{t}(x, g_z, y) \cdot \mathbf{t}(x, g_B, y) \Big] \end{aligned} \tag{26}$$

We know from Lemma 1 that this means $\mathbf{t}(x, g_B, y)$ should be a weighted superposition of $\mathbf{t}(x, g_L, y)$ and $\mathbf{t}(x, g_R, y)$. Since we know $\mathbf{e}_L = -\mathbf{e}_R$, we can write:

$$\begin{aligned} \mathbf{t}(x, g_L, y) &= (1-\delta) \cdot \mathbf{t}(x, y) + \mathbf{e}_L \\ \mathbf{t}(x, g_R, y) &= (1-\delta) \cdot \mathbf{t}(x, y) - \mathbf{e}_L \end{aligned}$$

Then the superposition of the two becomes $\mathbf{t}(x, y)$. But this is a semantically erroneous placement for $\mathbf{t}(x, g_B, y)$, as it will incorrectly be closer to any other instance of x and y co-appearing in a scene than the average $\mathbf{t}(x, g^{<rel>}, y)$. For example, for $g_A^{<rel>} = g_A$ where $f_{G,T}(g_A) =$ "_ above _", $\mathbf{t}(x, g_B, y) \cdot \mathbf{i}(x, g_A, y) > \mathbf{t}(x, g_L, y) \cdot \mathbf{i}(x, g_A, y)$ even though the two captions are equally inapplicable.

□

E.6. Contradiction for Condition 1 and 4

Now we show Condition 4 cannot be met if Condition 1 is met. Before we move to the proof, we first discuss in greater detail the possible arrangements of object concepts in C.

Orthogonality. One straightforward intuition is that since C is a high dimensional space, for any two random concepts $x^1, x^2 \in \mathbb{V}$, they should be approximately orthogonal [65]:

$$\mathbf{t}(x^j) \cdot \mathbf{t}(x^k) \approx 0 \quad \forall j \neq k \tag{27}$$

This makes it trivial to derive that $\mathbf{t}(\neg x) \cdot \mathbf{t}(y) \approx \mathbf{t}(x) \cdot \mathbf{t}(y) = 0$, violating Condition 4.2. In order to be more rigorous, we show that negation cannot be achieved even under strong idealistic conditions, where $\mathbf{t}(x)$ are uniformly distributed and *distinguishable* from one another. This requires perfect isotropy of M concepts.

Isotropy. Starting with $|\mathbb{V}| = M \leq N$ concepts, we determine the ideal distribution for $\mathbf{t}(x)$ $\forall x \in \mathbb{V}$. As in Lemma 1, we denote distinct concepts as: $\mathbf{t}(x^1), \mathbf{t}(x^2), \ldots$. Since all $x \in \mathbb{V}$ are mutually exclusive in semantics by definition, the distance between any two arbitrary concepts x^1, x^2 should be comparable to the distance between x^1, x^k for some $x^k \in \mathbb{V} \backslash \{x^2\}$. Then C must minimize the variance among the cosine similarities of all pairs of concepts:

$$\min_{\mathbf{t}(x^j)\cdot\mathbf{t}(x^k)} \sum_{j=1}^{M} \sum_{k>j}^{M} \left(\mathbf{t}(x^j) \cdot \mathbf{t}(x^k) - \bar{s} \right)^2 \quad \text{s. t. } \|\mathbf{t}(x^k)\| = 1, \quad \forall x^k \in \mathbb{V} \tag{28}$$

where $\bar{s}$ is the mean cosine similarity:

$$\bar{s} = \frac{1}{\binom{M}{2}} \sum_{j=1}^{M} \sum_{k>j}^{M} \mathbf{t}(x^j) \cdot \mathbf{t}(x^k) \tag{29}$$

Let $\mathbf{t}(x^j) \cdot \mathbf{t}(x^k) = s_{jk}$. Then the objective simplifies to:

$$\min_{s_{jk}} \sum_{j=1}^{M} \sum_{k>j}^{M} \left(s_{jk}^2 - \bar{s}^2 \right) = \min_{s_{jk}} \sum_{j=1}^{M} \sum_{k>j}^{M} (s_{jk})^2 - \bar{s}^2 \binom{M}{2} \tag{30}$$

Differentiate the first and second terms with respect to s_{jk}:

$$\frac{\partial}{\partial s_{jk}} \sum_{j=1}^{M} \sum_{k>j}^{M} s_{jk}^2 = 2 s_{jk}, \tag{31}$$

$$\frac{\partial}{\partial s_{jk}} \left(\bar{s}^2 \binom{M}{2} \right) = 2 \binom{M}{2} \bar{s} \frac{\partial \bar{s}}{\partial s_{jk}} = \frac{1}{\binom{N}{2}} 2 \binom{M}{2} \bar{s} = 2\bar{s} \tag{32}$$

This means that the optimum of Eq. (28) is reached when

$$2\bar{s} - 2 s_{jk} = 0 \quad \forall j,k \in (1,M), j \neq k \tag{33}$$

In other words, the cosine similarities between any two object concepts must be the same as the average. That requires for all $\mathbf{t}(x^k)$ to be isotropically distributed in $\mathbb{R}^N$. The optimal arrangement of all $\mathbf{t}(x^k)$ is then a M-1 dimensional regular simplex, which is a structure of M equiangular unit vectors in $\mathbb{R}^M$. Then we have that the cosine similarity of two random vectors in C is:

$$\boxed{\mathbf{t}(x^j) \cdot \mathbf{t}(x^k) = -\tfrac{1}{M-1} \quad \forall j \neq k} \tag{34}$$

as all vector pairs in a regular simplex have angles $\arccos(-\frac{1}{M-1})$ [47].

Lemma 5. Even under isotropic concept distribution, C cannot accurately represent negation.

Proof. We first derive what $\mathbf{t}(x), \mathbf{t}(\neg x)$ must be to respect Conditions 4.1 and 4.2. Then we see that this derivation contradicts Condition 4.3.

For C to ideally represent negation, the following must be true:

$$\begin{aligned} &\mathbf{t}(\text{"}\neg\, x\text{"}) \cdot \mathbf{i}(x) < \mathbf{t}(\text{"}\neg\, x\text{"}) \cdot \mathbf{i}(v) \quad (1) \\ &\mathbf{t}(\text{"}\neg\, x\text{"}) \cdot \mathbf{i}(v) > \mathbf{t}(\text{"}x\text{"}) \cdot \mathbf{i}(v) \quad (2) \\ &\mathbf{t}(\text{"}x\text{"}) \cdot \mathbf{t}(\text{"}y\text{"}) < \mathbf{t}(\text{"}\neg x\text{"}) \cdot \mathbf{t}(\text{"}\neg y\text{"}) \quad (3) \\ &\text{for all } v \in \mathbb{V} \backslash \{x\} \end{aligned} \tag{35}$$

To achieve Eq. (35.1), we solve for:

$$\begin{aligned} \mathbf{t}(\neg x) &= \underset{\mathbf{t}(\neg x)}{\text{argmax}} \left[\sum_{v=1}^{|\mathbb{V}\backslash\{x\}|} \mathbf{t}(\neg x) \cdot \mathbf{i}(v) - \mathbf{t}(\neg x) \cdot \mathbf{i}(x) \right] \\ &= \underset{\mathbf{t}(\neg x)}{\text{argmax}} \left[\mathbf{t}(\neg x) \cdot \left(\sum_{v=1}^{|\mathbb{V}|} \mathbf{i}(v) - \mathbf{i}(x) - \mathbf{i}(x) \right) \right] \end{aligned} \tag{36}$$

Here, $\sum_{v=1}^{|\mathbb{V}|} \mathbf{i}(v)$ is the sum of all vectors in a regular simplex, which is 0. As such, we find that:

$$\mathbf{t}(\neg x)^* = -\mathbf{i}(x) \tag{37}$$

Recall that for two vectors in an $M-1$ dimensional simplex, $\mathbf{t}(x^j) \cdot \mathbf{t}(x^k) = -\frac{1}{M-1}$ $\forall j \neq k$. With the above solution, we now have that:

$$\mathbf{t}(\neg x^j) \cdot \mathbf{t}(x^k) = \frac{1}{M-1} > \mathbf{t}(x^j) \cdot \mathbf{t}(x^k) \tag{38}$$

which satisfies condition 4.2. But 4.3 fails due to the following:

$$\boxed{\begin{aligned} &\mathbf{t}(\neg x^j) \cdot \mathbf{t}(\neg x^k) = \mathbf{t}(x^j) \cdot \mathbf{t}(x^k) = -\frac{1}{M-1} \\ &\mathbf{t}(\neg x^j) \cdot \mathbf{t}(x^k) > \mathbf{t}(\neg x^j) \cdot \mathbf{t}(\neg x^k) \end{aligned}} \tag{39}$$

Let's say $x^j =$ "chair" and $x^k =$ "penguin". The first erroneous semantic relationship that emerges is that the distance between "chair" and "penguin", which are fully contradictory statements, will be equivalent to the distance between "Not chair" and "Not penguin". For the latter two prompts there exist a lot of images that would be a true match for both, whereas for the first prompt there exists only one.

The second erroneous conclusion is that the cosine similarity between "Not chair" and "penguin" is greater than the cosine similarity between "Not chair" and "Not penguin". This is semantically incorrect for the same reason as above. □

F. Open Vocabulary Experimental Details

We train two model types, one on 5k COCO images from the training split for 12 epochs, and another on 10k COCO images for 6 epochs. The images each have a hard positive and negative, where the latter is the same as the former save for two nouns being swapped. We choose this particular type of intervention as CLIP-like VLM performance across the board was lowest for this category of hard negatives in

Original Prompts:

['A baby elephant stands next to its mother.', 'A desktop computer sitting on top of a wooden desk.', 'A mother stands next to its baby elephant.', 'A wooden computer sitting on top of a desk.']

Original Lookup Table:

['above', 'below', 'many', 'no', 'small', 'big', 'not', 'without', 'left', 'right', 'absent', 'but', 'large']

Simplified Prompts:

['Baby elephant next to mother', 'Desktop computer on wooden desk', 'Mother next to baby elephant', 'Wooden computer on desk']

New Lookup Table:

['above', 'below', 'many', 'no', 'small', 'big', 'not', 'without', 'left', 'right', 'absent', 'but', 'large', 'near', 'on']

New Prompts:

['Baby elephant ⟨NEAR⟩ mother', 'Desktop computer ⟨ON⟩ wooden desk', 'Mother ⟨NEAR⟩ baby elephant', 'Wooden computer ⟨ON⟩ desk']

Table 6. Examples of LLM-in-the-loop natural language simplification and FR extraction.

Sugarcrepe [16]. We evaluate this paradigm on the swap-object split of Sugarcrepe and the VG-spatial split of VL-Checklist [75] against naive and finetuned CLIP. We choose this particular split of VL-Checklist because we find that all other splits (Objects, Attributes, or Relation-Action) do not introduce new functional words and are thus not applicable for testing dynamic FR updates.

All LLM queries were made to OpenAI's gpt4o-mini model, with a temperature of 0.7. At the time of evaluation, this was the most affordable model on the API at: $ 0.150 / 1M input tokens and $0.600 / 1M output tokens.

F.1. LLM System Prompts

System prompts:

Given a list of sentences, reformat each sentence to the simplest phrases that would distinguish it from the other examples. Here are some formatting rules to follow:

1. If a sentence contains OBJECTS and ATTRIBUTES which belong to that object, the ATTRIBUTE must always come first. For example, given the sentence “A dog which is purple", reformat it to “A purple dog".

2. If a sentence contains NEGATION, the NEGATING TERM always comes before the OBJECT clause. For example, given “This image contains a chicken but a butterfly is absent", reformat it to “Chicken but no butterfly".

3. If a sentence contains PREPOSITIONs, try your best to make sure that the OBJECTS the PREPOSITION is describing are immediately before and after the PREPOSITION. For example, given “A bug which is flying much farther up from the bench", reformat it to “A flying bug above a bench".

4. Whenever possible, reformat VERBs to be ATTRIBUTES. For example, given “A dog dancing while his owner is jumping", reformat it to “Dancing dog and jumping owner".

5. If two sentences are very close to each other, reduce them down to the salient components. For example, given the sentences ["Butterflies in the clouds, a cat squatting looking up at it, and a man standing behind the cat watching it, on the grass with a tree.", “Butterflies in the clouds, a cat squatting looking up at it, and a man sitting behind the cat watching it, on the grass with a tree."], return: 0: “A man standing", 1: “A man sitting". (Of course, if there are other sentences in the list that are similar, you may want to keep more details so that the sentences are still distinguishable.)

Here are some more general examples. If given the following sentences: ["A desktop computer sitting on top of a gray oak table lights up the room" ,"A gray oak computer sitting on top of a desktop table lights up the room", “A kitchen has metal cabinets and black countertops with shiny lights on top.", “A kitchen has black cabinets and metal countertops with shiny lights on top."], return: 0: “Desktop computer top of gray oak table", 1: “Gray oak computer top of desktop table", 2: “Metal cabinets and black countertops", 3: “Black cabinets and metal countertops".

As a rule, be AS CONCISE AS POSSIBLE. If any information is repeated and unnecessary to keep in order to distinguish that text prompt from the others, discard it.

Now, reformat the following list of sentences and return the JSON output. Do not include anything other than the JSON array in your answer.

Table 7. Prompt template for simplifying natural language prompts.

System prompts:

You are given: A LOOKUP LIST of functional words (e.g., ["ABOVE", “BELOW", “INSIDE OF", “MANY", “SMALL", “NO"]). A list of SENTENCES to process. Definitions: Functional words include: (a) Prepositions (e.g., ABOVE, BELOW, INSIDE OF, ON, IN, NEAR) or their synonyms. (b) Size/shape terms (e.g., SMALL, BIG). (c) Numerical terms (e.g., ONE, TWO, THREE, etc.). If a number is greater than 5 (e.g., SEVEN, 100), replace it with “MANY". (d) Negatory terms (e.g., NO, WITHOUT). Non-functional words: Do not include verbs, adjectives, or any nouns unrelated to the functional categories above. Examples of non-functional words include “jumping", “sleeping", “cat", “man", etc. These should not be added to the LOOKUP LIST, even if they appear in the sentences. Even if there are two sentences that are very similar, do not try to distinguish them by adding these verbs, adjectives, or nouns to the LOOKUP LIST. Any form of a verb, including present participles, may not go in the LOOKUP LIST, no matter how frequently it appears in the sentences.

Rules: For each sentence: Identify any functional words or synonyms (including numbers). If a functional word or one of its synonyms (by meaning) appears in the sentence and is already in the LOOKUP LIST, replace it in the sentence with the LOOKUP LIST key, surrounded by angle brackets (e.g., “⟨ ABOVE⟩ “). If that functional word is not in the LOOKUP LIST (and it is truly functional by the above definition), add it to the LOOKUP LIST, then replace its appearance with that new all-caps key in angle brackets. Do not add duplicates to the LOOKUP LIST. Do not add verbs, adjectives, or any non-functional words to the LOOKUP LIST. Replace numbers greater than 5 with “MANY" (add “MANY" to the list if not already present). After processing all sentences, output exactly one JSON array containing two sub-arrays: The first sub-array: the UPDATED LOOKUP LIST (only functional words, no duplicates). The second sub-array: the FINAL TRANSFORMED SENTENCES (with functional words surrounded by ⟨ ⟩). Not every sentence needs functional words. Provide no additional commentary or text besides this JSON structure. Example of the required output format: [[“ABOVE", “INSIDE OF", “MANY" , “RIGHT OF"], [“A bird ⟨ ABOVE⟩ a tree", “Fifteen dogs is ⟨ MANY⟩ dogs" , “A sitting chicken is ⟨ INSIDE OF⟩ a house"]] Example of wrong output: [[“ABOVE", “INSIDE OF", “MANY" , “RIGHT OF", “SITTING", “DOGS", “HOUSE"], [“A bird ⟨ ABOVE⟩ a tree", “Fifteen dogs is ⟨ MANY⟩ dogs" , “A sitting chicken is ⟨ INSIDE OF⟩ a house"]]

Before you return the output, CHECK THAT THE LOOKUP LIST ONLY CONTAINS FUNCTIONAL WORDS.

Table 8. Prompt template for extracting functional words.

Swapping Objects:
Given an input sentence describing a scene, your task is to first locate two swappable noun phrases in the sentence, and then swap them to make a new sentence. The new sentence must meet the following three requirements: 1. The new sentence must be describing a different scene from the input sentence. 2. The new sentence must be fluent and grammatically correct. 3. The new sentence must make logical sense.

To complete the task, you should: 1. Answer the question of whether generating such a new sentence is possible using “Yes" or “No". 2. Output the swappable noun phrases. 3. Swap them to make a new sentence.

Please produce a **single JSON array** (no extra text or explanation) for each input sentence. If there are K input sentences, return a list with K JSON objects separated by commas. Each element in the array must be a JSON object with the following structure: { “possible": “⟨Yes or No⟩", “swappable-noun-phrases": ["⟨NP1⟩", “⟨NP2⟩"], “swapped-sentence": “⟨the swapped sentence⟩" }

Example JSON output for the original sentence: “A cat resting on a laptop next to a person." { “possible": “Yes", “swappable-noun-phrases": ["laptop", “person"], “swapped-sentence": “A cat resting on a person next to a laptop." }

Table 9. Prompt template for creating captions for swapped objects. This is a minorly changed version from [16].