Towards Extracting Text-Guided Image Attribute Representations

Supplementary Material

A. QARE-Bench Construction Details

In this section, we provide a detailed breakdown of the data generation and curation pipelines for QARE-BENCH.

A.1. Synthetic Set Generation

The core challenge in generating the synthetic set (4 Objects $\times$ 3 Styles $\times$ 4 Backgrounds) is *attribute consistency*. For example, when the attribute "Background" is fixed to "Forest", the generated forest must remain visually consistent (same lighting, flora, density) even when the "Object" changes from a person to a cat.

To achieve this *Semantic Locking*, we employed a **Dense Description Expansion** strategy using a commercial generation platform (Reve [1]). The process involves three steps:

1. **Base Prompting:** We start with a minimal tuple, e.g., *(Person, Forest, Photo)*.
2. **Dense Expansion:** We then expand this tuple into a highly detailed, paragraph-length description. This fixes variable details such as the direction of light, the texture of the ground, and the camera focal length.
3. **Template Slotting:** We identify the sentence clauses corresponding to each attribute and turn them into slots. By swapping only the content of specific slots while keeping the rest of the dense description frozen, we ensure that non-target attributes remain invariant.

An example of the dense prompt template structure and the resulting semantic locking effect is provided in Fig. 1.

Base Prompt Template

[Object Slot]: *A person faces forward, positioned in the center of the frame...*
[Background Slot]: *...stands in a dense forest. Dappled sunlight filters through the green canopy overhead, creating patches of light on the forest floor covered with brown pine needles. The trees' straight trunks extend upward...*
[Style Slot]: *...rendered in film-style photography with natural lighting, gentle grain, and sharp focus on the central subject.*

(a) Original prompt structure.

Modified Object Slot

[Object Slot] **(Changed):** *The dog faces forward, positioned in the center of the frame with the forest receding behind them.*
[Background Slot]: *(Locked / Same as left)*
[Style Slot]: *(Locked / Same as left)*

(b) Changing the object slot only.

(c) Person+Forest+Photo

(d) Dog+Forest+Photo

Figure 1. Semantic Locking in Synthetic Generation. To ensure strict attribute independence, we utilize dense description templates. (a) Shows the base prompt with fixed Background and Style slots. (b) Shows the modified prompt where only the Object slot is replaced. (c) and (d) demonstrate that the resulting images maintain semantic consistency in background and style, isolating the object as the sole variable.

A.2. Real Set Curation

The Real set is derived from a raw collection of high-resolution photographs where specific object instances were captured across multiple distinct scenes. The curation process is fully automated.

A.2.1. Object Query Pipeline

For the **Object** attribute, we employ a three-stage automated pipeline designed to create challenging queries that isolate the target object while retaining maximum visual context. We utilize Detic [2] as the base detector (confidence threshold > 0.45).

1. Context-Aware Dynamic Cropping. Unlike standard cropping which tightly bounds the object, we aim to include background context without introducing semantic ambiguity (i.e., other objects). For every detected object instance, we apply a **Dynamic Expansion** algorithm:

- **Expansion:** We initially expand the ground-truth bounding box by a factor of 3.0$\times$ to capture the surrounding environment.
- **Distractor Avoidance:** We detect all potential "distractor" objects within this expanded region. If any distractor overlaps with the target (IoU > 0.1), we iteratively shrink the crop boundary until it is tangent to the nearest distractor ('expand_until_touch').

This results in a *Dynamic Crop* that maximizes the visible background context while ensuring the target

remains the sole foreground subject.

2. Complexity-Driven Query Selection. To rigorously test the model's ability to disentangle the object from complex environments, we do not select the query image randomly. Instead, for a given unique object identity, we analyze all scenes in which it appears and select the scene with the **highest object density** (i.e., the maximum number of distractor objects) as the Query Scene. This forces the model to attend to the specific target amidst significant visual clutter.

3. Negative Mining. We construct negatives specifically to probe the model's reliance on background and context. Negatives are derived from two sources within the *same* query scene:

- **Background Negatives (Context-Only):** We extract the exact same crop region from the clean background plate (where the target object is physically absent or removed). These negatives share identical lighting and background pixels, differing only by the absence of the object.
- **Distractor Negatives (Semantic-Only):** We extract Dynamic Crops of *other* distractor objects present in the same query image. These negatives share the exact same global scene style, lighting condition, and semantic context, serving as hard negatives for fine-grained object discrimination.

A.2.2. Background Query Pipeline

For the *Background* attribute, the challenge is to find images where the *foreground objects* are identical (to serve as hard negatives) but the *scene* is different. We implement the following pipeline:

1. Object Grouping. We first identify groups of interacting objects within a scene. Objects are clustered into a group if their expanded bounding boxes ($1.05\times$) overlap. A "Group Crop" is defined as the union of the bounding boxes of all objects in the cluster.

2. Query Selection. For each group, we select the image that contains the most frequent appearance of the constituent objects as the **Query Image**. We enforce a minimum crop size of 100×100 pixels.

3. Spatial Alignment (Key Step). To generate valid hard negatives (same objects, different background), we locate the same set of objects in a *different* scene. We calculate the relative position of the objects within the source group crop and project this geometry onto the target scene. This ensures that the resulting crop in the new scene preserves the exact composition (scale and relative position) of the foreground objects, isolating the background as the only changing variable.

4. Similarity-Based Filtering. Finally, to ensure ground-truth correctness, we verify the background consistency using a visual similarity metric (ResNet50 features):

- **Positives:** Crops from the same scene must share high background similarity (Threshold > 0.6).
- **Negatives:** Crops from different scenes must have low background similarity (Threshold < 0.4) while containing the same foreground objects.

Groups that fail to meet minimum counts (e.g., at least 2 positives and 3 negatives) are discarded.

B. TF-QARE Implementation Details

In this section, we provide the exact prompt templates used for all attributes, describe the memory-efficient inference strategy, and detail the specific configurations for the VLM backbones used in our experiments.

B.1. Structured Prompt Templates

As introduced in the main paper, TF-QARE utilizes structured prompts to disentangle visual attributes. While the *Object* prompt was presented in the main text, we provide the exact templates for *Background* and *Style* in Fig. 2. Consistent with the object prompt, these templates enforce a strict output format (Main Summary + Detailed Description) and utilize negative constraints to prevent attribute leakage.

B.2. Memory-Efficient Two-Stage Inference

A naive implementation of feature extraction—requesting hidden states during autoregressive generation—can be computationally prohibitive. In standard HuggingFace Transformers implementations, setting `output_hidden_states=True` during `generate()` forces the model to return full hidden states for all layers at every decoding step. This incurs significant memory overhead due to repeated tensor allocation and aggregation.

To address this, we implement a memory-efficient **Two-Stage Inference** scheme:

1. **Stage 1: Text Generation.** We perform a standard optimized forward pass to generate the reply text $\mathcal{R}$ for each (image, prompt) pair. During this stage, we utilize Key-Value (KV) caching and disable hidden state outputs to maximize throughput.
2. **Stage 2: Feature Extraction.** We concatenate the original input prompt and the generated reply $\mathcal{R}$ into a single sequence. We then perform a *single* batched forward pass on this complete sequence to

Prompt for Background Extraction

Input Command:
Describe ONLY the background environment using a two-part structured format.
FORMAT MUST MATCH EXACTLY:
Background is [main summary], [detailed description].

Rules:
- [main summary]: 1–3 words summarizing the overall scene or setting.
- [detailed description]: 15–30 words adding visual details about lighting, weather, elements, or depth.
- Do NOT mention the main object or any stylistic terms.
- Write up to TWO sentences; no lists, no line breaks, no quotes.

Prompt for Style Extraction

Input Command:
Describe ONLY the visual style (tone, color, lighting, texture, composition, mood) using a two-part structured format.
FORMAT MUST MATCH EXACTLY:
Visual style is [main summary], [detailed description].

Rules:
- [main summary]: 1–3 words summarizing the overall tone or color scheme.
- [detailed description]: 15–30 words expanding on lighting, texture, color balance, and mood.
- Do NOT mention specific objects, people, animals, actions, locations, background, environment, or scene.
- Write up to TWO sentences; no lists, no line breaks, no quotes.

Figure 2. Full structured prompts for *Background* and *Style* attributes used in TF-QARE.

extract the hidden states corresponding to the reply tokens.

This approach decouples generation from extraction, allowing us to process the extraction step in parallel batches without the step-wise overhead of the autoregressive loop.

B.3. VLM Configurations

We implement TF-QARE using PyTorch 2.6 and the HuggingFace Transformers library. All experiments were conducted on NVIDIA A100 (80GB) GPUs.

To ensure stable and reproducible embeddings, we employ **greedy decoding** (temperature = 0) for all experiments. This eliminates randomness in the generated text, ensuring that the subsequent feature extraction yields consistent representations for the same input. Detailed inference hyperparameters are listed in Table 1.

Table 1. Inference configurations used for all experiments.

Hyperparameter	**Value**
Decoding Strategy	Greedy Search (`do_sample=False`)
Temperature	0
Top-p / Top-k	N/A (Greedy)
Max New Tokens	64
Precision	bfloat16 (BF16)
Repetition Penalty	1.0 (Default)

C. Qualitative Results

In this section, we present qualitative comparisons between TF-QARE and the baseline VLM2Vec-v2. Both methods utilize the **Qwen2-VL-7B** backbone. The retrieval database consists of embeddings for every possible (Image, Attribute) pair. In all figures:

- **Leftmost Column:** Displays the **Query Image** and the **Target Attribute** (e.g., Object, Background).
- **Retrieved Columns:** Display the Top-10 retrieved candidates.
- **Headers:** Indicate (1) Rank; (2) Cosine Similarity; and (3) The Attribute associated with that candidate embedding.
- **Color Coding:** **Green** indicates a correct match (Ground Truth); **Red** indicates an error.

C.1. Analysis on Synthetic Set

As shown in Fig. 3, TF-QARE demonstrates exceptional sensitivity to the queried attribute while remaining invariant to others. For **Object** queries (Rows 1–3), our method retrieves the target object across diverse backgrounds (e.g., "Car in Forest" retrieving "Car in Room"), effectively filtering out scene information. Conversely, for **Background** queries (Rows 4–6), it ignores the salient foreground object and retrieves scenes with the identical environment but different objects.

In contrast, the baseline VLM2Vec (Fig. 4) exhibits significant *attribute entanglement*. It frequently suffers from "background leakage" during object queries (e.g., retrieving a "Dog in Forest" when querying for a "Car in Forest"), driven by the strong visual signal of the background. Furthermore, the baseline often retrieves

candidates with incorrect attribute types (e.g., returning a "style" embedding for an "object" query), indicating a failure to follow the prompt instructions.

References

[1] Reve. Reve image generation platform. https://app.reve.com. Accessed: 2025-11-20. 1

[2] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. In *ECCV*, 2022. 1

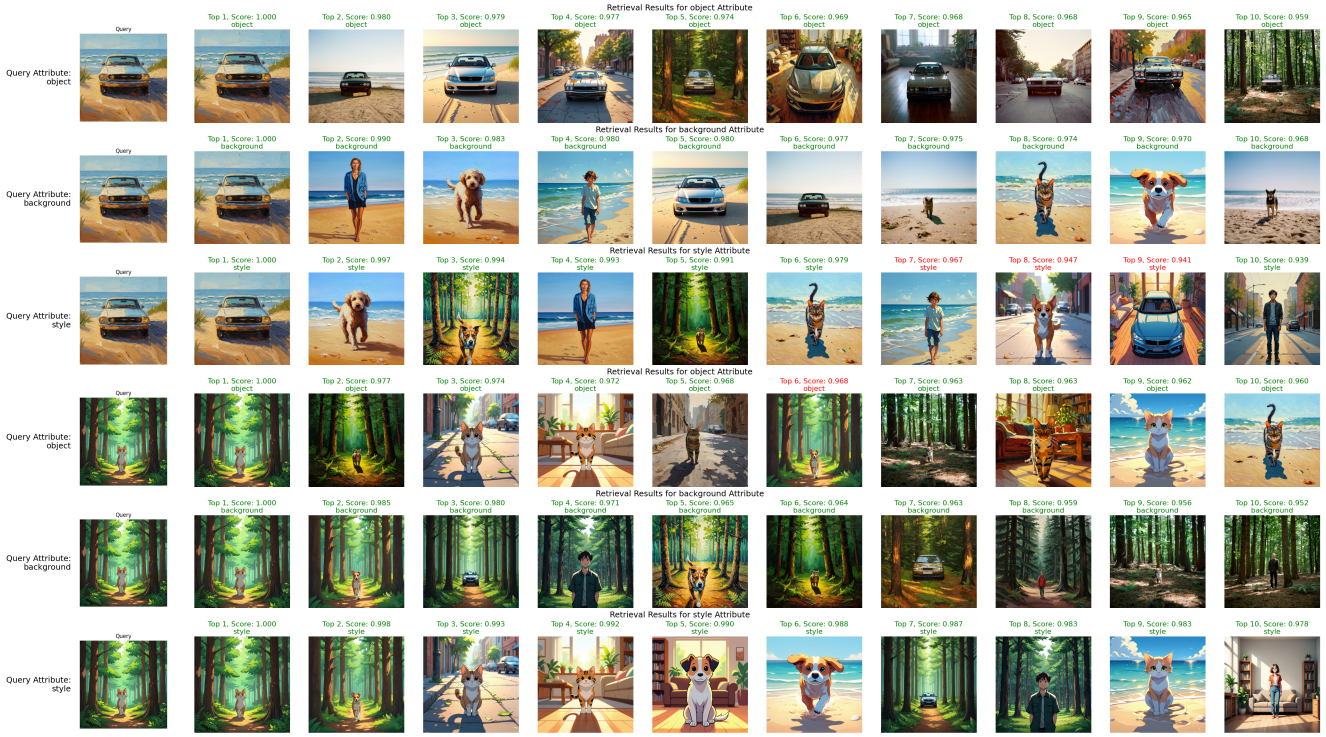

Figure 3. Qualitative Results on QARE-Bench Synthetic Set (Method: TF-QARE).

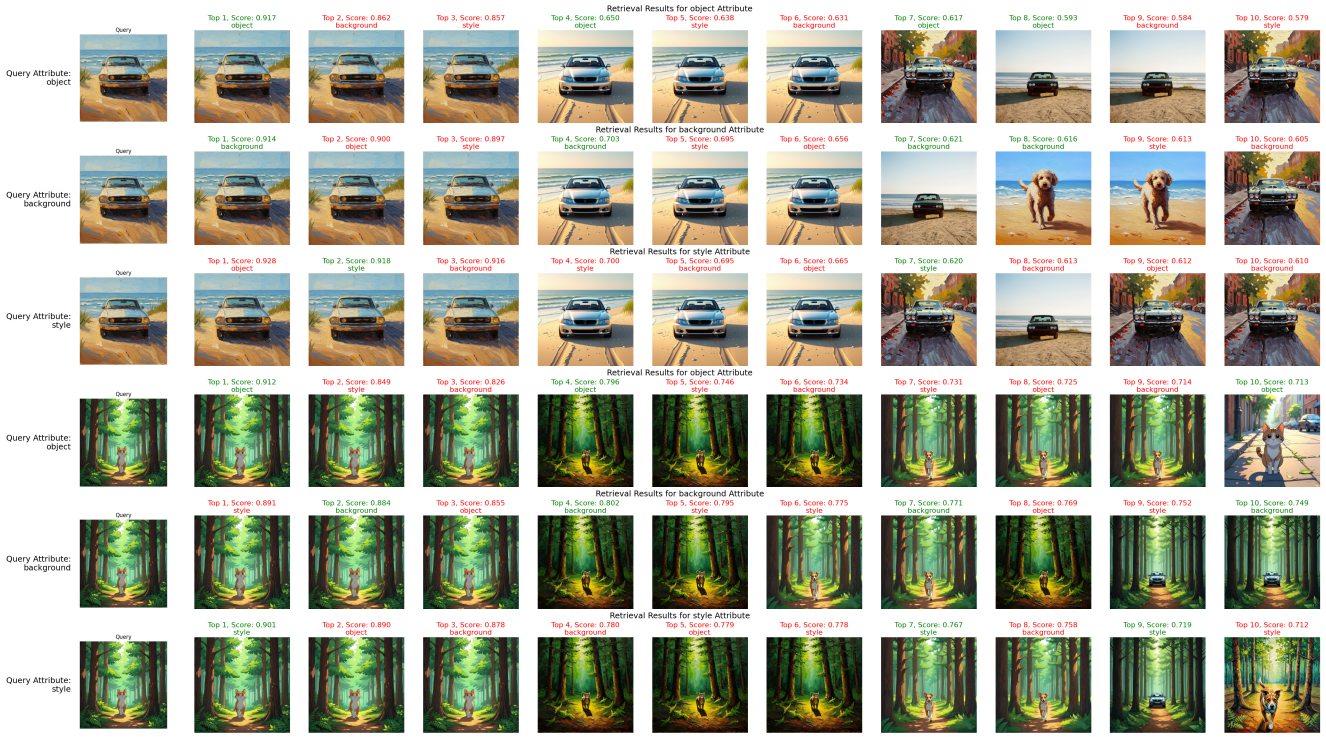

Figure 4. Qualitative Results on QARE-Bench Synthetic Set (Method: VLM2Vec).

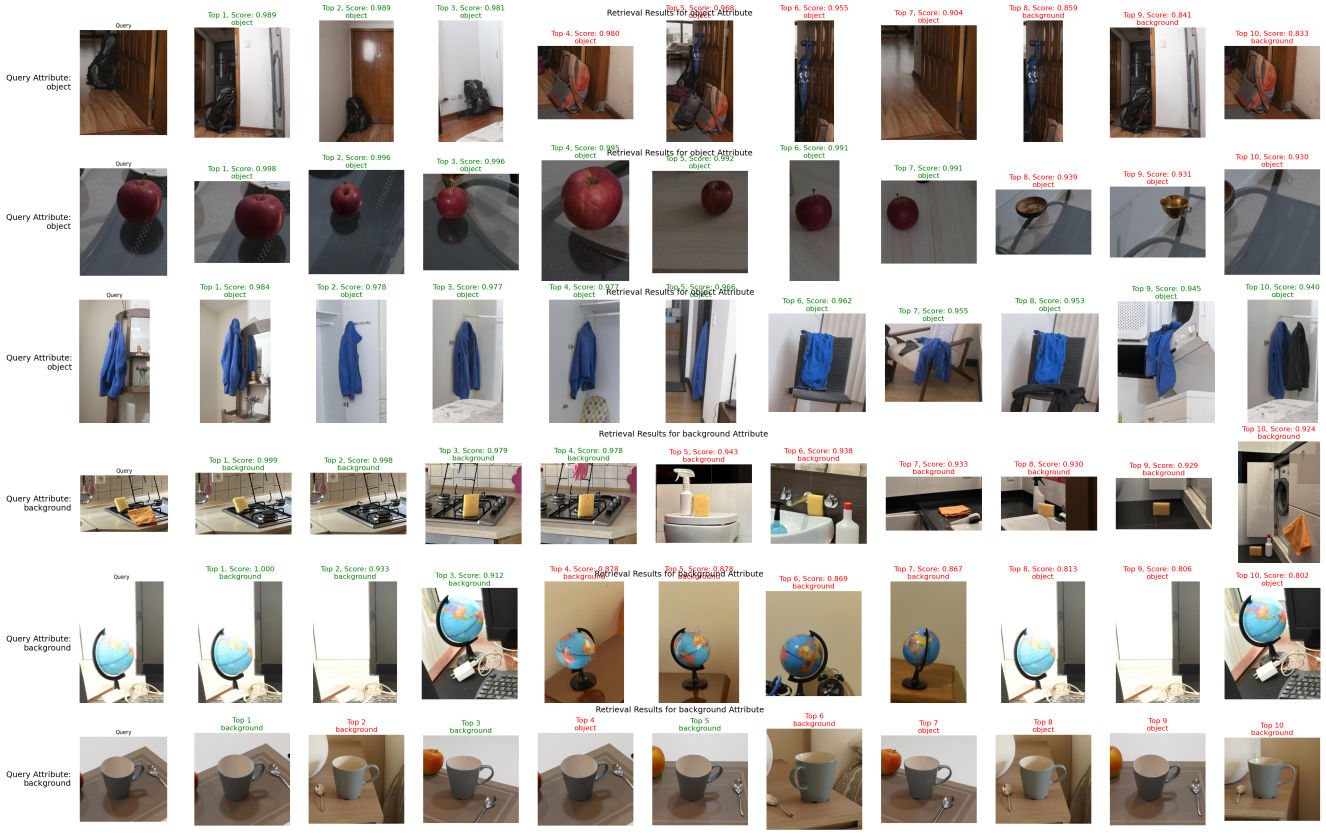

Figure 5. Qualitative Results on QARE-Bench Real Set (Method: TF-QARE).

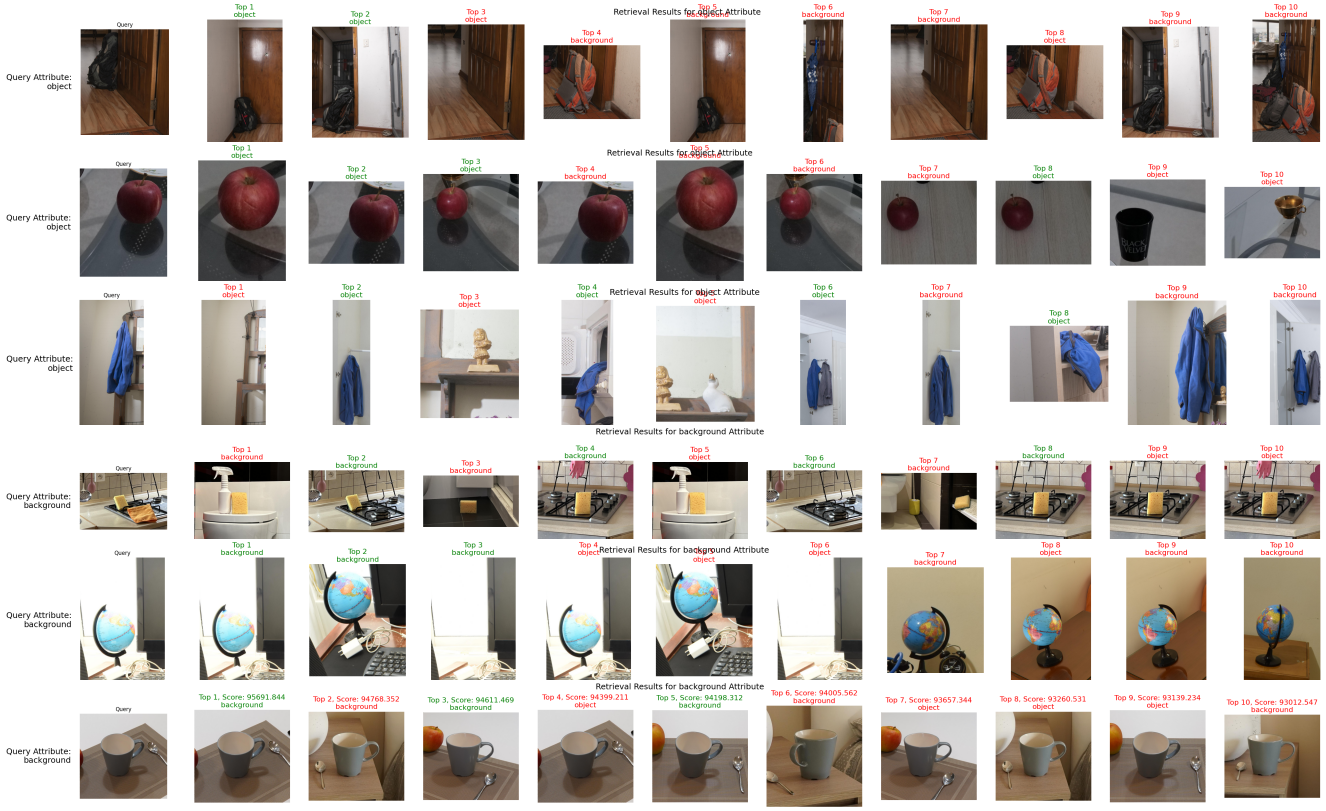

Figure 6. Qualitative Results on QARE-Bench Real-World Set (Method: VLM2Vec).