STATS 302 Project Report Draft

Topic: CLIP retrieval failure analysis + targeted text-side improvement
Model: `openai/clip-vit-base-patch32`
Dataset: MS COCO 2017 Val captions/images
Evaluation: Image retrieval from text; Recall@K (R@1 / R@5 / R@10)
Seed: 42 (for subset sampling + runs)

> Figures are left as placeholders (to be added later).

---

1. Overview

We study the failure modes of a CLIP-based text-to-image retrieval system and test a lightweight improvement method that does **not** retrain the model. The workflow is:

1. Run baseline CLIP retrieval and cache embeddings on a fixed subset of COCO val.
2. Sample retrieval failures (R@1 failures) for manual annotation.
3. Categorize failures using a taxonomy (Ambiguous / Object / Attribute / Action / Context / Count / Spatial).
4. Use the annotated subsets to evaluate **prompt ensembling** with different **template pooling** strategies.
5. Compare baseline vs improved performance on each error category.

---

2. Experimental Setup

2.1 Subset construction & caching

We sample `NUM_IMAGES = 5000` images from COCO val and take up to `CAPTIONS_PER_IMAGE = 5` captions per image (fixed caption order). We then compute and cache:

- image embeddings: `img_5000_openai_clip-vit-base-patch32.pt`
- text embeddings: `txt_5000_openai_clip-vit-base-patch32.pt`
- metadata: `meta_5000_openai_clip-vit-base-patch32.json`
- captions list: `captions_5000_openai_clip-vit-base-patch32.json`

Baseline retrieval results on the 5000-image subset (from run logs):

- R@1 = **0.3044**
- R@5 = **0.5477**
- R@10 = **0.6624**

Command (baseline + cache):

```
python main-v2.py
```

Implementation reference: the caching + baseline evaluation logic is in `main_re.py` (v2 variant used in runs).

---

3. Failure Sampling + Annotation Pipeline

3.1 Failure sampling (R@1 failures)

We compute similarity `sim = txt_emb @ img_emb.T`, take top-1 retrieved image per caption, and collect all captions where:

- `top1_idx[i] != gt_img_index[i]`

From the baseline run, we found:

- **17,389** R@1 failures (on the 25,000 captions of the 5000-image subset).

We then sample **200** failures using `seed=42` to make the annotation set reproducible.

3.2 Assignment design (A/B/C + overlap)

From the sampled 200 failures:

- overlap set: 30 items (shared to estimate agreement / consistency)
- each annotator: 50 items (A/B/C)
- remaining: backup set

The script exports:

- `failure_analysis/failure_samples.csv` (raw sample)
- `failure_analysis/assign_overlap.csv`
- `failure_analysis/assign_A.csv`
- `failure_analysis/assign_B.csv`
- `failure_analysis/assign_C.csv`
- optional visuals under `failure_analysis/vis_*`

Command (sampling + assignment + visuals):

Implementation reference: assignment generation and visualization are in `failure-v2.py`.

```
python failure-v2.py
```

---

4. Failure Taxonomy Analysis

4.1 Merged A/B/C (excluding overlap)

After annotating and merging A/B/C labels (excluding overlap), the category distribution is:

category_clean	count	percent
Ambiguous	67	31.90
Object	62	29.52
Attribute	25	11.90
Action	24	11.43
Context	16	7.62
Count	15	7.14
Spatial	1	0.48

Key observation.

- A large fraction is **Ambiguous (31.9%)**, suggesting many failures are not easily correctable via simple text-side modifications.
- Among actionable categories, the dominant ones are **Object (29.52%)**, then **Attribute (11.90%)** and **Action (11.43%)**.
  This motivates focusing improvements on actionable subsets (excluding Ambiguous).

4.2 Figures (placeholders)

- **Figure 1.** Overall category distribution (all labels)
  (placeholder)
- **Figure 2.** Actionable-only distribution (excluding Ambiguous)
  (placeholder)

---

5. Improvement Method: Category-Aware Prompt Ensembling

5.1 Motivation

CLIP retrieval depends on text embeddings. Many failures appear related to how captions specify:

- main objects
- attributes (color/texture/style)
- actions
  Thus, we test whether expanding each caption into multiple paraphrased templates and aggregating them improves retrieval rank.

5.2 Method

For each caption in a target subset (e.g., Object), generate K templated prompts. Encode each prompt using CLIP text encoder; compute similarity to cached image embeddings; then pool similarity scores across templates.

Pooling strategies

- **max**: take maximum similarity over templates (per image)
- **mean**: average similarity over templates (per image)
- **logsumexp** (soft pooling): soft-max-like aggregation with temperature `tau=1.0`

Command template (subset evaluation):

```
python improve_subset.py \
  --annotations_csv .\failure_analysis\analysis_all\annotations_clean.csv \
  --include_categories <Category or Category1,Category2> \
  --max_subset 200 \
  --do_prompt_ensemble \
  --pooling <max|mean|logsumexp> \
  --tau 1.0
```

Implementation reference: improvement evaluation is implemented in `improve_subset.py`.

---

6. Results (Per-Category Improvements)

We report baseline vs improved Recall@K on annotated subsets (seed=42).
All values are percentages (%). Deltas are percentage points (pp).

6.1 Action (n=24, K=10)

Baseline: R@1 **0.00**, R@5 **33.33**, R@10 **45.83**

pooling	improved R@1	improved R@5	improved R@10	ΔR@1 (pp)	ΔR@5 (pp)	ΔR@10 (pp)
max	4.17	37.50	66.67	+4.17	+4.17	+20.83
mean	4.17	33.33	66.67	+4.17	+0.00	+20.83
logsumexp (tau=1.0)	4.17	33.33	66.67	+4.17	+0.00	+20.83

Interpretation.

- Large improvement at **R@10 (+20.83pp)** across pooling methods: prompt variants help push correct images into top-10.
- R@5 improvement is sensitive to pooling (max helps more here), but **n=24 is small**, so variance is expected.

Figure 3 (placeholder): Action subset: baseline vs improved (R@K) across pooling.

6.2 Attribute (n=25, K=10)

Baseline: R@1 **0.00**, R@5 **44.00**, R@10 **52.00**

pooling	improved R@1	improved R@5	improved R@10	ΔR@1 (pp)	ΔR@5 (pp)	ΔR@10 (pp)
max	4.00	44.00	52.00	+4.00	+0.00	+0.00
mean	0.00	48.00	52.00	+0.00	+4.00	+0.00
logsumexp (tau=1.0)	0.00	48.00	52.00	+0.00	+4.00	+0.00

Interpretation.

- Attribute failures are harder: improvements mainly appear at **R@5** (+4pp), while **R@10 stays unchanged**.
- Mean/logsumexp perform better than max on R@5, suggesting attribute-focused prompts benefit from more stable pooling rather than relying on a single best template.

Figure 4 (placeholder): Attribute subset: baseline vs improved across pooling.

6.3 Object (n=62, K=10)

Baseline: R@1 **0.00**, R@5 **33.87**, R@10 **51.61**

pooling	improved R@1	improved R@5	improved R@10	ΔR@1 (pp)	ΔR@5 (pp)	ΔR@10 (pp)
max	3.23	37.10	53.23	+3.23	+3.23	+1.61
mean	1.61	37.10	54.84	+1.61	+3.23	+3.23
logsumexp (tau=1.0)	1.61	38.71	54.84	+1.61	+4.84	+3.23

Interpretation.

- Object failures show consistent gains.
- **logsumexp is best overall** (largest R@5 and tied-best R@10), indicating soft pooling balances max/mean behavior.

Figure 5 (placeholder): Object subset: baseline vs improved across pooling.

6.4 Object + Attribute (n=87, K=15)

Baseline: R@1 **0.00**, R@5 **36.78**, R@10 **51.72**

pooling	improved R@1	improved R@5	improved R@10	ΔR@1 (pp)	ΔR@5 (pp)	ΔR@10 (pp)

pooling	improved R@1	improved R@5	improved R@10	ΔR@1 (pp)	ΔR@5 (pp)	ΔR@10 (pp)
max	2.30	40.23	52.87	+2.30	+3.45	+1.15
mean	1.15	43.68	52.87	+1.15	+6.90	+1.15
logsumexp (tau=1.0)	1.15	43.68	52.87	+1.15	+6.90	+1.15

Interpretation.

- This is the **largest actionable subset** and most representative of common failures.
- **mean/logsumexp doubles the R@5 gain** vs max (+6.90pp vs +3.45pp).
- With larger K (=15), max pooling may be more sensitive to noisy templates; mean/logsumexp are more robust.

Figure 6 (placeholder): Object+Attribute subset: baseline vs improved across pooling.

---

7. Discussion

7.1 What improves, and why

- **Object** and **Object+Attribute** subsets benefit most: many errors are due to under-specified or differently phrased object descriptions; prompt variants capture alternate lexicalizations.
- **Action** benefits mainly at R@10: multiple templates help bring action-consistent images into the top-10 even if top-5 ordering remains challenging.
- **Attribute** is difficult: fine-grained attribute discrimination often requires visual detail beyond what templating can reliably add.

7.2 Pooling matters

Across categories, pooling controls stability:

- For mixed categories and larger K, **mean/logsumexp** are more robust than max (notably on Object+Attribute).
- For single-category Object, **logsumexp** performs best overall (largest R@5 and strong R@10).

7.3 Limitations

- Several subsets are small (Action n=24, Attribute n=25), so results may have high variance.
- **Ambiguous** failures (31.9%) are inherently hard: underspecified captions and near-duplicate images likely require additional context or alternative evaluation protocols rather than text-side templating.

---

8. Recommended Configuration (Based on Ablations)

A single robust improvement setting for actionable failures:

- Use **category-aware templates**

- Use **logsumexp pooling** with **tau = 1.0** (stable across Object-heavy subsets; tied-best on mixed subset)

This choice yields:

- Object: best R@5 and strong R@10
- Object+Attribute: best R@5 (tied with mean)
- Attribute: improves R@5 (tied with mean)

---

9. Reproducibility: Commands Summary

9.1 Baseline + cache

```
python main-v2.py
```

9.2 Failure sampling + assignment (seed=42 fixed inside script)

```
python failure-v2.py
```

9.3 Analyze annotation CSV (per file or merged)

```
python analyze_annotations.py --csv
<assign_A.csv|assign_B.csv|assign_C.csv|assign_all.csv> --outdir <output_dir>
```

9.4 Improvement runs (examples)

Object+Attribute (pooling ablation):

```
python improve_subset.py --annotations_csv
.\failure_analysis\analysis_all\annotations_clean.csv --include_categories
Object,Attribute --max_subset 200 --do_prompt_ensemble --pooling max
python improve_subset.py --annotations_csv
.\failure_analysis\analysis_all\annotations_clean.csv --include_categories
Object,Attribute --max_subset 200 --do_prompt_ensemble --pooling mean
python improve_subset.py --annotations_csv
.\failure_analysis\analysis_all\annotations_clean.csv --include_categories
Object,Attribute --max_subset 200 --do_prompt_ensemble --pooling logsumexp --
tau 1.0
```

---

10. Appendix (Placeholders)

A.1 Figures to add

- Fig 1: category distribution (overall)
- Fig 2: actionable-only distribution
- Fig 3–6: per-subset baseline vs improved across pooling (bar chart or line)

A.2 Tables to add

- Consolidated table of all runs (the CSV summary you generated)