CLIP-ZeroShot-Retrieval

This repository provides a **reproducible pipeline (for core metrics and result tables)** for analyzing **failure modes of a pre-trained vision–language model (CLIP)** on image–text retrieval using the MS-COCO dataset.

The project focuses on:

- Zero-shot image–text retrieval
- Systematic failure taxonomy annotation
- Analysis of ambiguous vs. semantic failures
- Failure-driven improvement experiments (prompting / re-ranking)

The pipeline is designed for **beginners** and does **not require model training or fine-tuning**.

Project Overview

Model

- CLIP (OpenAI), zero-shot
- `openai/clip-vit-base-patch32`

Dataset

- MS-COCO 2017 (validation split)
- Image–caption pairs

Key Idea
Rather than only reporting retrieval accuracy, we:

1. Identify Top-1 retrieval failures
2. Annotate failures with a structured taxonomy
3. Separate *ambiguous* errors from *clear semantic failures*
4. Analyze which failure types dominate
5. Explore targeted improvements on specific failure subsets

Repository Structure

Layout

```
.
├── src/                    # core python scripts
├── scripts/                # batch runners
├── outputs/
│   ├── cache/              # embeddings + frozen metadata (generated)
│   ├── subset_results/     # per-subset experiment json outputs
```

```
│   ├── subset_hits/            # per-sample hit csv outputs
│   ├── summary/                # summarized csv outputs
│   └── figures/                # generated figures
├── configs/                    # run configs (e.g., baseline_1k.yaml)
├── data/                       # COCO images + annotations
├── docs/                       # guides, definitions, baseline/improvement
notes (+ pdf exports)
├── failure_analysis/           # annotation CSVs, analysis scripts,
visualizations
├── requirements.txt
```

Migration Note

This reorganization has been applied with `proposals/` and `trash file/` left unchanged.

Environment Setup

1. Python Environment

Python ≥ 3.9 is recommended.

```
conda create -n clip_failure python=3.9
conda activate clip_failure
```

2. Install Dependencies

```
pip install torch torchvision torchaudio
pip install transformers
pip install pillow tqdm
pip install pycocotools
```

or you can refer to the `requirements.txt`

```
pip install -r requirements.txt
```

GPU is optional but recommended.

Dataset Preparation

Download MS-COCO 2017 (Validation)

1. Images
   https://cocodataset.org/#download
   Download **2017 Val images**

   which is: http://images.cocodataset.org/zips/val2017.zip

2. Annotations
   Download **captions_val2017.json**

   which is: http://images.cocodataset.org/annotations/annotations_trainval2017.zip

3. Backup:
   Download from Duke box: https://duke.box.com/s/c3kdhyenkrjs8eh6ua5ks1276p1qrta1

Place files as:

```
data/
├── val2017/
│   ├── 000000000139.jpg
│   └── ...
└── annotations/
    └── captions_val2017.json
```

Quickstart (reproduce main results)

A) Fast path (already have `annotations_clean.csv`)

Use this when `failure_analysis/analysis_all/annotations_clean.csv` already exists.

1. Run all improvement experiments + summary + bootstrap CI:

```
scripts\experiment_run.bat
```

2. Expected outputs:

- `outputs/subset_results/subset_results_*.json`
- `outputs/subset_hits/subset_hits_*.csv`
- `outputs/summary/summary_subset_results.csv`

3. Success check:

- Terminal prints `[FINISH] Done`
- `outputs/summary/summary_subset_results.csv` is generated/updated

B) Full path (include merge + analysis)

Use this when teammates finished manual labels (`assign_A/B/C/overlap.csv`) and need a full post-annotation pipeline.

1. Run:

```
scripts\post_annotations_run.bat
```

2. This script will do:

- `src/merge_assignments.py` → merged assignment files
- `failure_analysis/analyze_annotations.py` → `failure_analysis/analysis_all/annotations_clean.csv`
- `src/improve_subset.py` (Object+Attribute / Object / Attribute / Action)
- `src/summarize_result.py` → `outputs/summary/summary_subset_results.csv`
- `src/bootstrap_ci.py` (if Action hits csv exists)

3. Success check:

- Terminal prints `[FINISH] Done`
- `failure_analysis/analysis_all/annotations_clean.csv` exists
- `outputs/summary/summary_subset_results.csv` exists

C) Optional pre-step (rebuild cache from scratch)

If teammates need to regenerate embeddings/failure samples from raw COCO:

```
python src/main.py
python src/failure.py
```

Expected outputs:

- `outputs/cache/img_5000_*.pt`, `txt_5000_*.pt`, `meta_5000_*.json`, `captions_5000_*.json`
- `failure_analysis/assign_*.csv`, `failure_analysis/vis_*/`

D) One-page verification checklist

After running A or B, verify:

- `outputs/subset_results/` has JSON files for Action / Attribute / Object / Object-Attribute
- `outputs/subset_hits/` has Action hit CSVs (`max`, `mean`, `logsumexp`)
- `outputs/summary/summary_subset_results.csv` contains pooling columns (`max`, `mean`, `logsumexp`)
- `outputs/figures/` contains report figures (`fig_pooling_*`, `fig_best_*`, optional `fig_bootstrap_*`)
- Running `python src/summarize_result.py` again finishes without errors

0) Setup

```
conda create -n clip_failure python=3.9
conda activate clip_failure
pip install -r requirements.txt
```

Step 1: Run Baseline Retrieval

`src/main.py` performs:

- Random sampling of 5,000 images
- Extraction of image & text embeddings
- Caching for reproducibility
- Recall@K evaluation

```
python src/main.py
```

Output

- Cached embeddings in `outputs/cache/`
- Metadata with frozen ordering (`meta_*.json`)
- Console output:

```
R@1  = 0.30
R@5  = 0.55
R@10 = 0.66
```

> ⚠ Do NOT delete cache files unless you intend to recompute embeddings.

Step 2: Extract Retrieval Failures

`src/failure.py`:

- Identifies Top-1 failures
- Samples 200 failures
- Splits tasks across annotators
- Saves visualizations (GT vs Retrieved)

```
python src/failure.py
```

Output

- CSV task files:

  - `assign_overlap.csv`
  - `assign_A.csv`
  - `assign_B.csv`
  - `assign_C.csv`

- Visualization folders:

  - `vis_overlap/`
  - `vis_A/`, `vis_B/`, `vis_C/`

Each image shows:

- Left: Ground Truth
- Right: Top-1 Retrieved
- Caption at top

Step 3: Manual Annotation (Human-in-the-Loop)

How to Annotate

See: `docs/annotations_guide_en.md` (or `docs/annotations_guide_cn.md`).

1. Open your assigned CSV (e.g., `assign_A.csv`)

2. For each row:

   - Open `fail_{idx}.jpg` in your visualization folder
   - Read the caption
   - Compare GT vs Retrieved

3. Fill in:

```
category (Ambiguous(underspecified, nearduplicate, the subtypes are optional);
Attribute; Action; Count; Context; Sptial; Object.)
ambiguous_subtype   (only if category = Ambiguous)
```

Allowed Categories

See `docs/annotations_guide_en.md` for exact definitions.

For labelling, to make it easier, we use shorter version:

```
Ambiguous (underspecified, nearduplicate)
Attribute
Action
Count
```

```
Context
Spatial
Object
```

Here is the more detailed version

```
Attribute Binding
Object Confusion
Spatial Relation
Action / Interaction
Scene / Context
Counting / Plurality
Ambiguous
```

Annotation Rules (Summary)

- One label per sample
- If unsure → Ambiguous
- Do NOT over-interpret small visual differences

Step 4: Merge Annotations & Analyze

After annotation:

- Merge CSVs: run `python src/merge_assignments.py`

- Compute:

  - Category distribution
  - Ambiguous vs clear failure ratio
  - Inter-annotator agreement (overlap set)

Step 5: Summarize results (tables/plots-ready CSV)

```
python src/summarize_result.py
```

Typical findings:

- A large fraction of failures are **Ambiguous**
- Clear failures cluster around **Attribute** and **Object**

Step 6: Failure-Driven Improvements

This repository supports **lightweight improvements without training**:

```
python src/improve_subset.py
```

Examples

- Prompt ensembling for attribute-heavy captions
- Re-ranking using lexical cues (TF-IDF / caption overlap)
- Category-specific prompting

Evaluation should be:

- **Subset-based** (only on clear failures)

Reproducibility Notes

- All randomness is seed-controlled
- Embedding order is frozen via `meta_*.json`
- Cached embeddings ensure consistent results
- CSV indices map directly to embedding rows
- Reproducibility should be judged by `subset_results/*.json`, `subset_hits/*.csv`, and `summary/summary_subset_results.csv`
- Figure files (`png`/`pdf`) may differ at binary level across environments (e.g., matplotlib/font/backend), while numeric results remain consistent

---

Outputs

After running the pipeline, you should see:

- `outputs/cache/`
    - image/text embeddings + `meta_*.json` (frozen order)
- `failure_analysis/`
    - `assign_overlap.csv`, `assign_A.csv`, ...
    - `vis_overlap/`, `vis_A/`, ...
- `outputs/summary/` (and other outputs folders)
    - `summary_subset_results.csv` (used for report plots)

Note: delete `outputs/cache/` only if you want to recompute embeddings.

Key configs (defaults)

`seed`: controls the sampled subset + any random splits `subset_size`: number of failure cases per category

`pooling`: {mean, max, logsumexp} for template aggregation

`K_templates`: number of prompt templates

`tau`: temperature for logsumexp (if used)

(Where these are set: src/main.py / src/improve_subset.py)

Failure taxonomy labels

Main label (category) must be one of:

- Ambiguous, Attribute, Action, Count, Context, Spatial, Object

If `category=Ambiguous`, optionally fill:

ambiguous_subtype in {underspecified, nearduplicate}

Intended Audience

This project is suitable for:

- Undergraduate ML courses
- First research projects in multimodal learning
- Students learning how to analyze model failures

No prior experience with large-scale ML training is required.

Citation & Acknowledgments

- CLIP: Radford et al., *Learning Transferable Visual Models From Natural Language Supervision*
- MS-COCO Dataset

License

This project is for **educational and research purposes only**.