Assignment 1:
Image Retrieval via Multimedia Encoding

Due: 11:59 pm Central Time on Sept 23, 2022.

# Goal

In this assignment, you will develop an image retrieval system using use multimedia embeddings. Given the image, it should be able to retrieve corresponding captions/text descriptions, and vice versa. The main idea is to represent each image and each caption using multimedia encoders, and find the captions (or images) most similar to a query image (or caption).

# Data

1. MSCOCO dataset:

**\*IMPORTANT\*\***

**Note that we currently accept both 2014 version and 2017 version of the COCO dataset.**

Each image is associated with five text descriptions. We use the data split following Karpathy split settings (Karpathy & Fei-Fei, 2015). Please download the images at <https://cocodataset.org/#download> using 2014 version, and follow the split anntoations in <https://cs.stanford.edu/people/karpathy/deepimagesent/caption_datasets.zip>

**\*\*IMPORTANT\*\* Detailed Instructions of Processing MSCOCO 2014 version:**

1. Download the zip files for MSCOCO images (2014 version):
	1. Train: <http://images.cocodataset.org/zips/train2014.zip>
	2. Val: <http://images.cocodataset.org/zips/val2014.zip>
	3. Test: <http://images.cocodataset.org/zips/test2014.zip>
2. Unzip these image files and got three folders of images: train2014, val2014, test2014.
3. Download the dataset split files at: <https://cs.stanford.edu/people/karpathy/deepimagesent/caption_datasets.zip>. Unzip the file and got a folder with three json files: *dataset\_coco.json*, *dataset\_flickr8k.json*, *and dataset\_flickr30k.json*.
4. Here we only use ***dataset\_coco.json****,* which is the split file for MSCOCO dataset. Read the json file into a Python dictionary and then go into the *“images”* dict key. In each *“image”* object, you can use “filepath” and “filename” to point back to the path of the original image.

For example, if image[“filename”]=”val2014” and image[“filepath”]=” COCO\_val2014\_000000391895.jpg”, then the path of this image would be val2014/ COCO\_val2014\_000000391895.jpg.

1. In each image, you can use the “split” dict key to identify which split the image belongs to. There could be four possible values in “split”: [“train”, “val”, “test”, “restval”]. **Note that all “restval” images should also be included into the “train” split, which will result in 113,287 train images, 5,000 val images, and 5,000 test images.**

1. Flickr30K dataset:
Each image is associated with five text descriptions. Flickr30K contains 31,000 images, we also use the same data split as [1], where there are 29,000 training images, 1000 test images, and 1000 validation images. Please download the images at <https://www.kaggle.com/datasets/hsankesara/flickr-image-dataset>. The caption annotations are at <https://github.com/BryanPlummer/flickr30k_entities/blob/master/annotations.zip>. The train/val/test splits are at <https://github.com/BryanPlummer/flickr30k_entities/blob/master/train.txt> , val.txt, test.txt.

# 3 Code Resources

* + CLIP <https://github.com/openai/CLIP>
	+ BLIP <https://github.com/salesforce/BLIP>
	+ ALBEF <https://github.com/salesforce/ALBEF>
	+ ALPRO <https://github.com/salesforce/ALPRO>
	+ Huggingface <https://huggingface.co/docs/transformers/model_doc/vit>

# 4 Evaluation Code

Image Retrieval is commonly evaluated with Average Precision (AP) or Recall@k. In this homework, we will compute Recall@1, Recall@5 and Recall@10 for both image-to-text (i2t) retrieval and text-to-image (t2i) retrieval.

The example code is as follows.

|  |
| --- |
| def i2t(image\_len, sims, npts=None, return\_ranks=False): """ Images->Text (Image Annotation) Images: (N, n\_region, d) matrix of images Captions: (5N, max\_n\_word, d) matrix of captions CapLens: (5N) array of caption lengths sims: (N, 5N) matrix of similarity im-cap """ npts = image\_len ranks = np.zeros(npts) top1 = np.zeros(npts) for index in range(npts): inds = np.argsort(sims[index])[::-1] # Score rank = 1e20 for i in range(5 \* index, 5 \* index + 5, 1): tmp = np.where(inds == i)[0][0] if tmp < rank: rank = tmp ranks[index] = rank top1[index] = inds[0] # Compute metrics r1 = 100.0 \* len(np.where(ranks < 1)[0]) / len(ranks) r5 = 100.0 \* len(np.where(ranks < 5)[0]) / len(ranks) r10 = 100.0 \* len(np.where(ranks < 10)[0]) / len(ranks) medr = np.floor(np.median(ranks)) + 1 meanr = ranks.mean() + 1 if return\_ranks: return (r1, r5, r10, medr, meanr), (ranks, top1) else: return (r1, r5, r10, medr, meanr)def t2i(image\_len, sims, npts=None, return\_ranks=False): """ Text->Images (Image Search) Images: (N, n\_region, d) matrix of images sims: (N, 5N) matrix of similarity im-cap """ npts = image\_len ranks = np.zeros(5 \* npts) top1 = np.zeros(5 \* npts) # # --> (5N(caption), N(image)) # sims = sims.T for index in range(npts): for i in range(5): inds = np.argsort(sims[5 \* index + i])[::-1] ranks[5 \* index + i] = np.where(inds == index)[0][0] top1[5 \* index + i] = inds[0] # Compute metrics r1 = 100.0 \* len(np.where(ranks < 1)[0]) / len(ranks) r5 = 100.0 \* len(np.where(ranks < 5)[0]) / len(ranks) r10 = 100.0 \* len(np.where(ranks < 10)[0]) / len(ranks) medr = np.floor(np.median(ranks)) + 1 meanr = ranks.mean() + 1 if return\_ranks: return (r1, r5, r10, medr, meanr), (ranks, top1) else: return (r1, r5, r10, medr, meanr) |

# 5 Submission

Please name your submission as `netid\_assignment1.zip`, including a report named as `netid\_report.pdf` and the code `netid\_code`. The code should include a README.md with environment and running instructions.

Please submit your assignment on Canvas: <https://canvas.illinois.edu/courses/31904/assignments/569506>

# Tasks

1. **(5pt)** Apply any multimedia encoding model to get image retrieval results on MSCOCO.
2. **(4pt)** Evaluate and report the evaluation scores of your implementation, including Recall@1, Recall@5 and Recall@10 for both image-to-text retrieval and text-to-image retrieval. The grades will be assigned based on the rank of your system among all systems from the class.
3. **(3pt)** Writeclear and informative written report about your methods, results and findings.
4. **(3pt)** Writeclear README of the submitted code, including environment and running instructions. Please submit **your source code including a shell script named “run\_pred.sh”** that can run evaluation on the test set. If you have finetune/train the model, please include those commands into the shell script.
5. **(Bonus, 2pt)** Apply the model on another dataset Flickr30k and compare the results on two datasets. Detail your observations from the comparison.
6. **(Bonus, 5pt)** Improve your system (simple finetuning will not count). For example, leverage the entity/event information to improve the retrieval results. The grades will be assigned based on the rank of your improved system among all systems from the class.