ELSEVIER

HAYSTACK

Learning a Joint Embedding Representation for Image Search using Self-supervised means

Haystack Conference 2022

April 2022

Sujit Pal (ORCID ID: https://orcid.org/0000-0002-6225-110X)

Who am I?

- Work for Elsevier Labs
- Technology Research Director
- Areas of interest: Search, NLP, ML
- Interested in improving search through Machine Learning

Agenda

- Background
- Model fine-tuning
- Image Search
- Demos

Background

Motivation

- Image Search
- Existing Approaches
  - Text search on image captions
  - Searching on image tags
- Embedding based search
- Unsupervised or self-supervised

The Scream is the popular name given to a composition created by Norwegian Expressionist artist Edvard Munch in 1893. The agonised face in the painting has become one of the most iconic images of art, seen as symbolising the anxiety of the human condition.

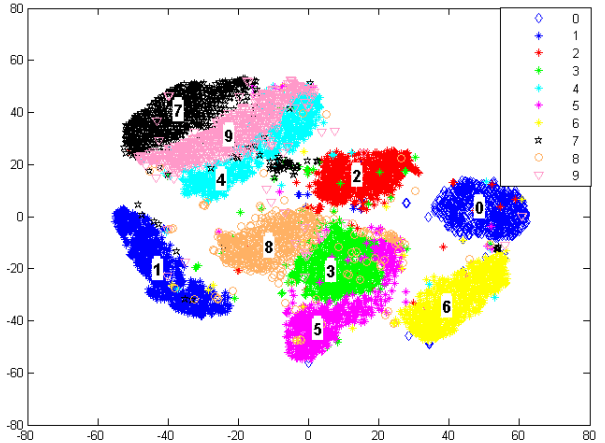
Embeddings: Origin Story

- **Distributional Hypothesis** – words that are used or occur in similar contexts tend to purport similar meanings [*Wikipedia*]
- **Word2Vec** – neural network learns weights that project words from a sparse high dimensional vocabulary space to a dense low dimensional (300-D) space.
- Words that have similar meaning tend to cluster together in this space.

Image Credits: [Explainable Artificial Intelligence (Part II) – Model Interpretation Strategies](#) (Sarkar, 2018)

Embeddings for Image Search

- Same clustering effect is observed with images as well
- Unsupervised: MNIST images encoded using Sparse AutoEncoders and projected to 2D show clustering behavior.
- Supervised: Images encoded using pre-trained ImageNet networks and encodings used as input to downstream classifiers. Needs (relatively) small amounts of training data for task.



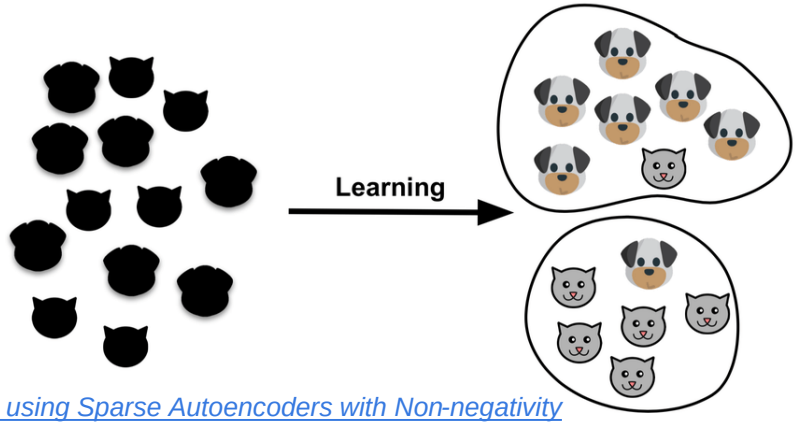


Image Credits: *Deep Learning of Part-based Representation of Data using Sparse Autoencoders with Non-negativity constraints (Hosseini et al, 2015),* *Learning a multi-view weighted majority vote classifier (Goyal, 2018)*

The need for Contrastive Learning

- Image Search == Image Similarity, i.e., a ranking problem
- Relevant results must appear above less-relevant results
- Classification Loss (cross-entropy) better suited to learning hyperplanes that separate one class of images from another.
- Contrastive Loss better suited for ranking images by similarity
- Pairwise loss – embedding pushes positive pairs close together and negative pairs far apart.

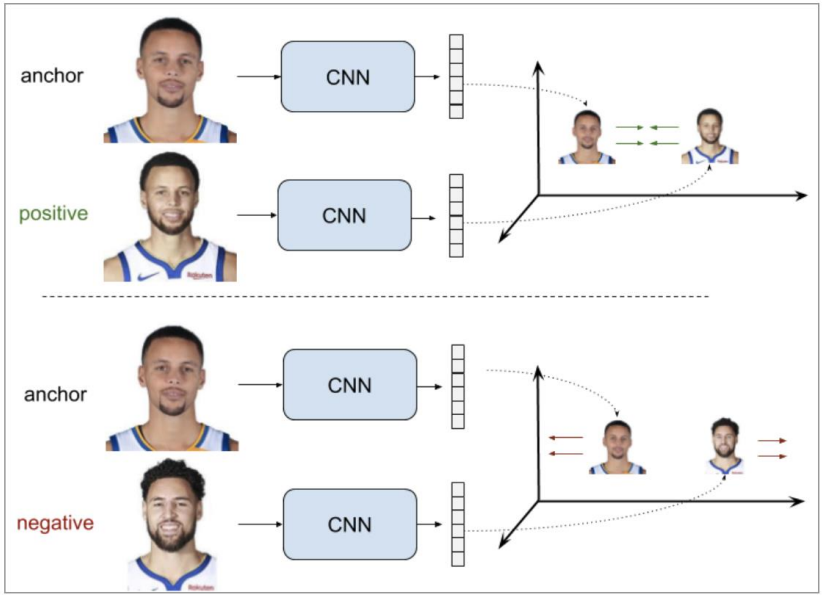


Image Credit: [Understanding Ranking Loss, Contrastive Loss, Margin Loss, Hinge Loss and all those confusing names](#) (blog post by Raul Gomez)

Contrastive Learning

- Uses Contrastive Loss instead of Cross-Entropy
- Given a pair of items x_0 and x_1 and a label y (1 if x_0 and x_1 are similar, 0 otherwise), contrastive loss is defined as:

$$\mathcal{L}(x_0, x_1, y) = y \, \|x_0 - x_1\| + (1 - y) max(0, m - \|x_0 - x_1\|)$$

- m is the minimum marginal loss for negative pairs. If the margin is already high for dissimilar pairs, no additional effort is wasted trying to push them further apart.

Advantages of Contrastive Learning

- No explicit labels
- Model learns from pairs of similar or dissimilar items
- Only similar pairs need to be provided, dissimilar pairs can generally be inferred
- Pairs can be multi-modal – for example, text + audio, text + video, and text + images.
- In our case (images + captions)

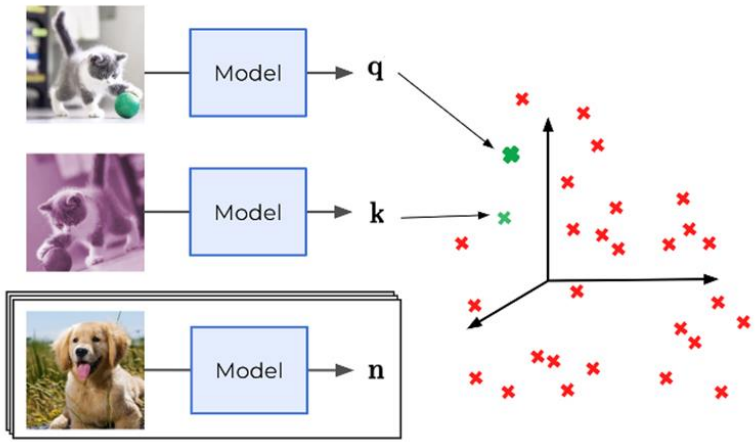
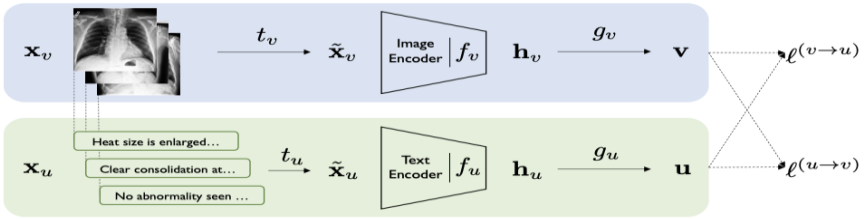
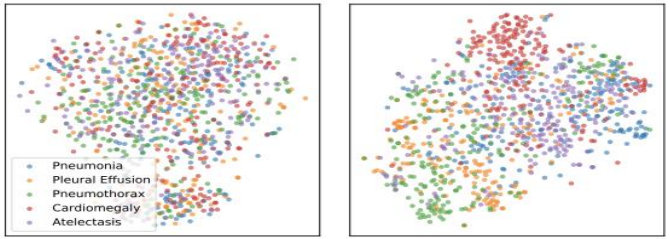


Image Credit: [Improving self-supervised representation learning by synthesizing challenging negatives](#) (Kalantidis et al, 2020)

Previous Work: ConVIRT

- My inspiration for this work.
- Trained on medical images and captions (self-supervised)
- Siamese network
- Image Encoder: ResNet50
- Text Encoder: BERT
- Minimizes bi-directional loss between positive image-text pairs
- ConVIRT image encoders uniformly did better on downstream tasks compared to previous approaches.



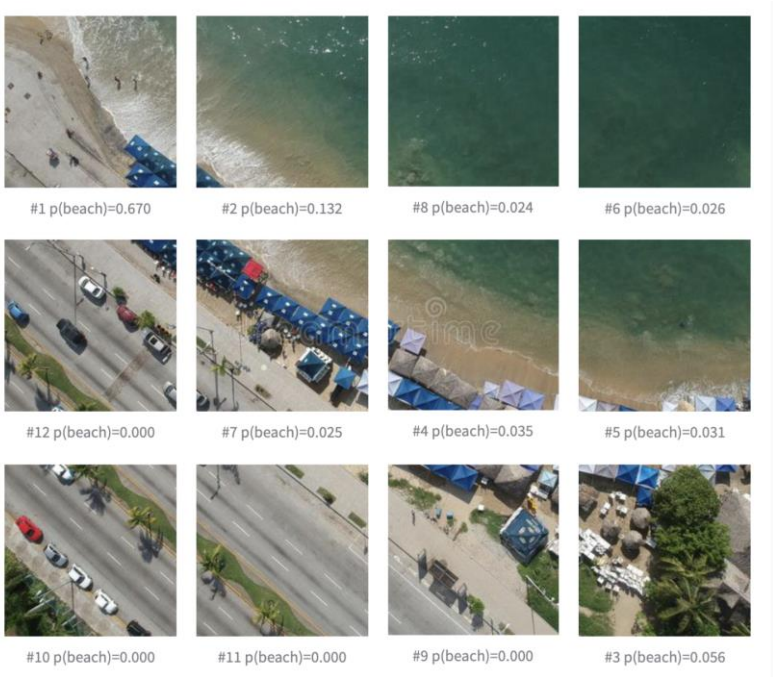


(a) ImageNet Pretraining (b) ConVIRT Pretraining

Image Credit: [Contrastive Learning of Medical Visual Representations from Paired Images and Texts](#) (Zhang et al, 2020)

Fine tuning CLIP for a different domain

- Part of HuggingFace🤗 JAX/Flax community week event
- Part of team of ”non-work colleagues” from TWIML
- Meant to train on medical images but another team was quicker, so trained on satellite images instead
- Placed third
- That’s how I learned about CLIP
- Blog post has more details



Model Fine-tuning

Our Base Model: OpenAI CLIP

- Contrastive Language Image Pretraining
- Pretrained model, trained in self-supervised manner on 400M image-caption pairs from the Internet
- Trained on 256 GPUs for 2 weeks
- Matches original ResNet50 performance on ImageNet
- Good zero-shot performance on ”natural images”
- Architecture:
  - Text Encoder: Text Transformer
  - Image Encoder: Vision Transformer
- Available on HuggingFace🤗

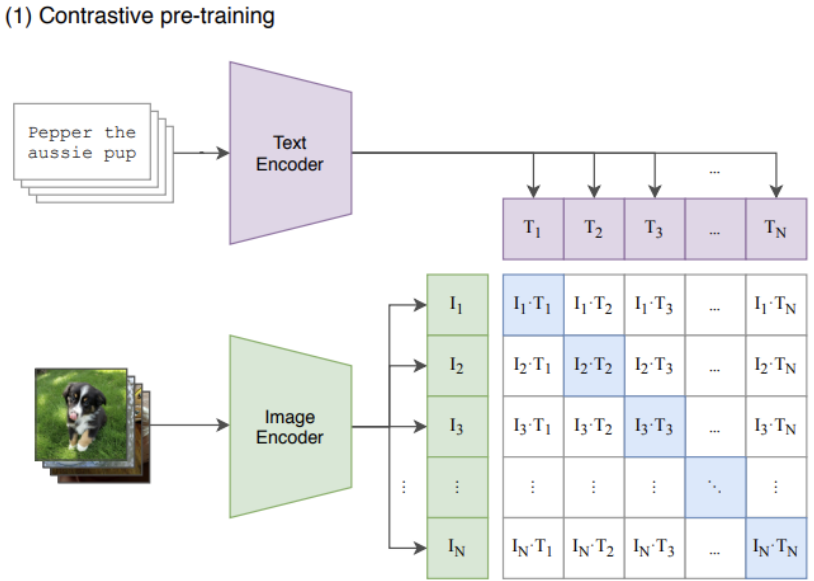


Image Credit: [Learning Transferable Visual Models from Natural Language Superviseion](#) (Radford et al, 2021)

Fine-tuning CLIP

- Fine-tuning == extending the contrastive pretraining with additional image + caption pairs from the target domain
- For each batch of size *N*:
  - *N* positive pairs
  - N^2 – *N* negative pairs
- Trained using Contrastive Objective against positive pairs and sampling from in-batch negative pairs.

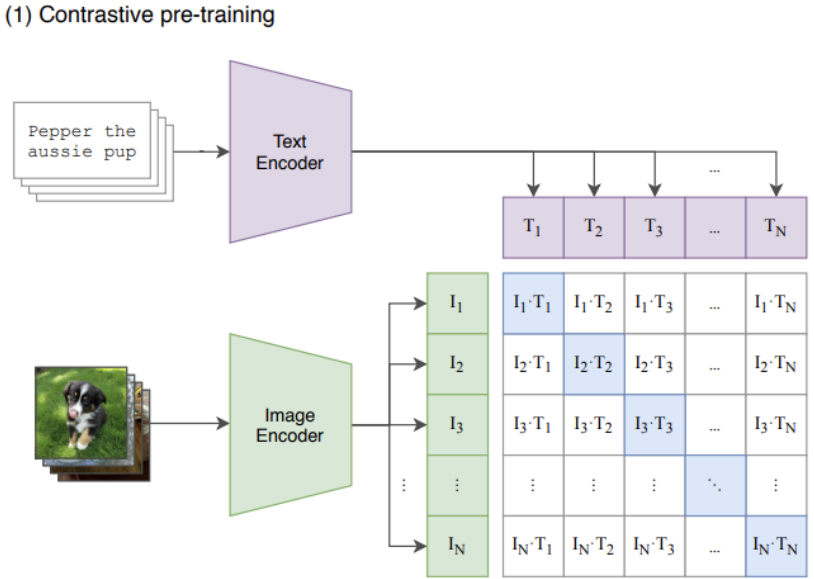
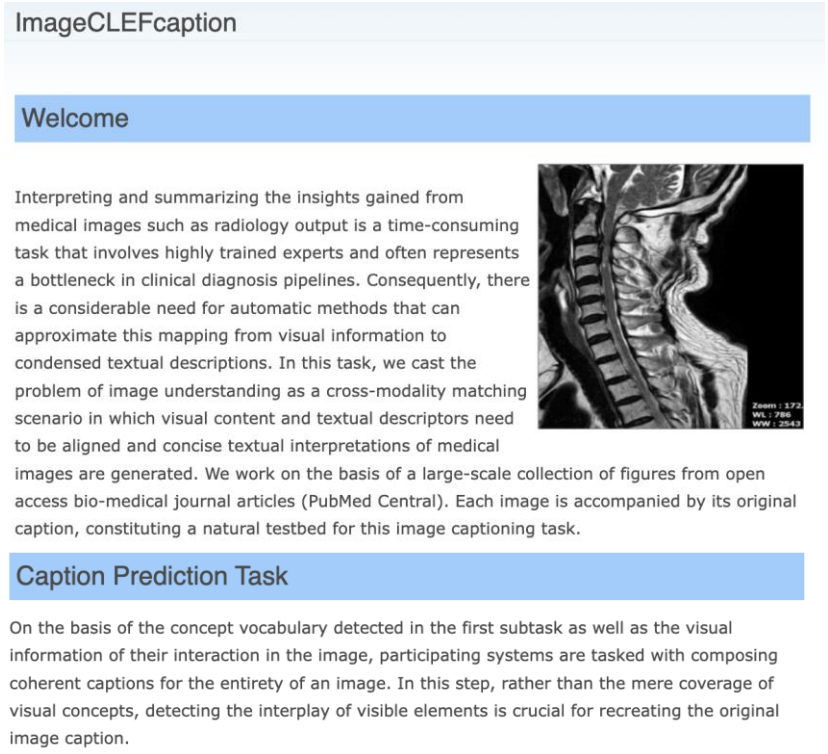


Image Credit: [Learning Transferable Visual Models from Natural Language Superviseion](#) (Radford et al, 2021)

Dataset

- ImageCLEF 2017 Caption Prediction Task
  - 164,614 training images and captions
  - 10,000 validation images and captions
  - 10,000 test images (without caption)
- Repurposed training and validation sets for fine-tuning on medical images + captions.



ImageCLEFcaption

Welcome

Interpreting and summarizing the insights gained from medical images such as radiology output is a time-consuming task that involves highly trained experts and often represents a bottleneck in clinical diagnosis pipelines. Consequently, there is a considerable need for automatic methods that can approximate this mapping from visual information to condensed textual descriptions. In this task, we cast the problem of image understanding as a cross-modality matching scenario in which visual content and textual descriptors need to be aligned and concise textual interpretations of medical images are generated. We work on the basis of a large-scale collection of figures from open access bio-medical journal articles (PubMed Central). Each image is accompanied by its original caption, constituting a natural testbed for this image captioning task.

Caption Prediction Task

On the basis of the concept vocabulary detected in the first subtask as well as the visual information of their interaction in the image, participating systems are tasked with composing coherent captions for the entirety of an image. In this step, rather than the mere coverage of visual concepts, detecting the interplay of visible elements is crucial for recreating the original image caption.

Dataset Preprocessing

- Ignore provided test split for training
- Split training dataset 90:10 into training and validation
- Make the validation split the new test split
- Final data split
  - 148,153 training images + captions
  - 16,461 validation images + captions
  - 10,000 test images + captions

Evaluation

- **Criteria:** Mean Reciprocal Rank (MRR) @k

$$MRR@k = \frac{1}{k} \sum_{i=1}^{k} \frac{1}{rank_i}$$

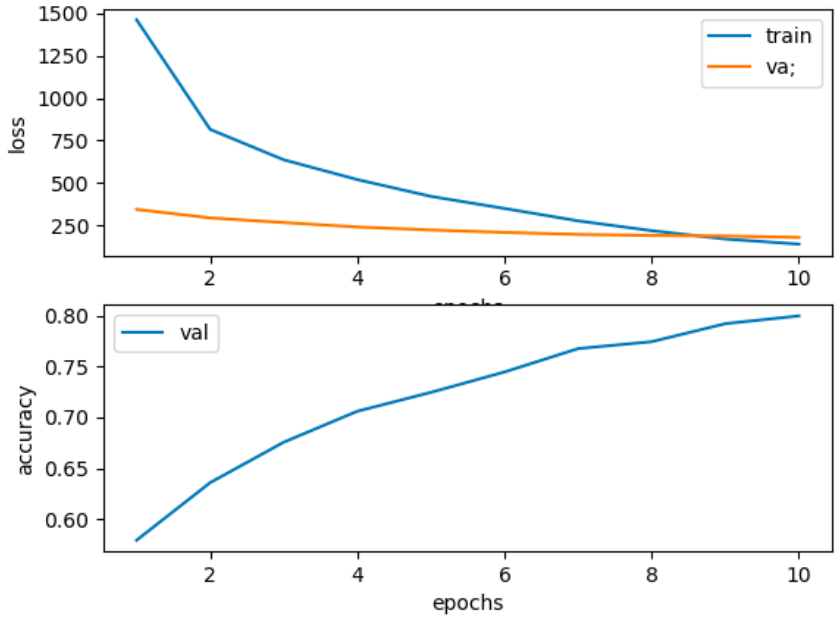
- **Baseline:** evaluated pre-trained CLIP model OOB (before fine-tuning)

Experiment	k=1	k=3	k=5	k=10	k=20
baseline	0.42580	0.53402	0.55837	0.57349	0.57829

- Correct caption for image returned 43% of the time at first position, 57% of the time within the first 10 positions.

Model Training

- Best training hyperparameters
  - Batch size 64
  - Optimizer ADAM
  - Learning Rate 5e-5
  - Number of epochs 10
  - Number of training samples 50,000
- Regularization achieved by sampling random subset of 50k samples from 148k training dataset
- No image or text augmentation
- Training loss still falling, but validation loss flattening out, maybe still some room for improvement



Training Results

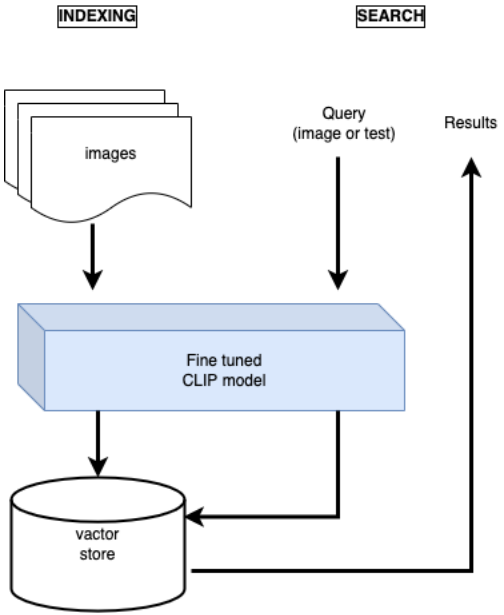
- Evaluated for multiple runs against the (new) test set.
- Results show significant improvement over baseline.

Experiment	k=1	k=3	k=5	k=10	k=20
baseline	0.42580	0.53402	0.55837	0.57349	0.57829
run-1	0.69130	0.78962	0.80113	0.80517	0.80589
run-2	0.71200	0.80445	0.81519	0.81912	0.81968
run-3	0.34540	0.46338	0.49253	0.51154	0.51753
run-4	0.78760	0.86227	0.86870	0.87080	0.87120
run-5	**0.80200**	**0.87170**	**0.87743**	**0.87966**	**0.88002**

Image Search

Image Search Architecture

- Encode images into L2-normalized vectors using fine-tuned model
- Store encoded image vectors in vector store
- At query time, query text or image is encoded using same fine-tuned model
- Vector store searched for encodings that are “close” to the query vector
- Images and captions corresponding to the top K closest vectors returned in query results

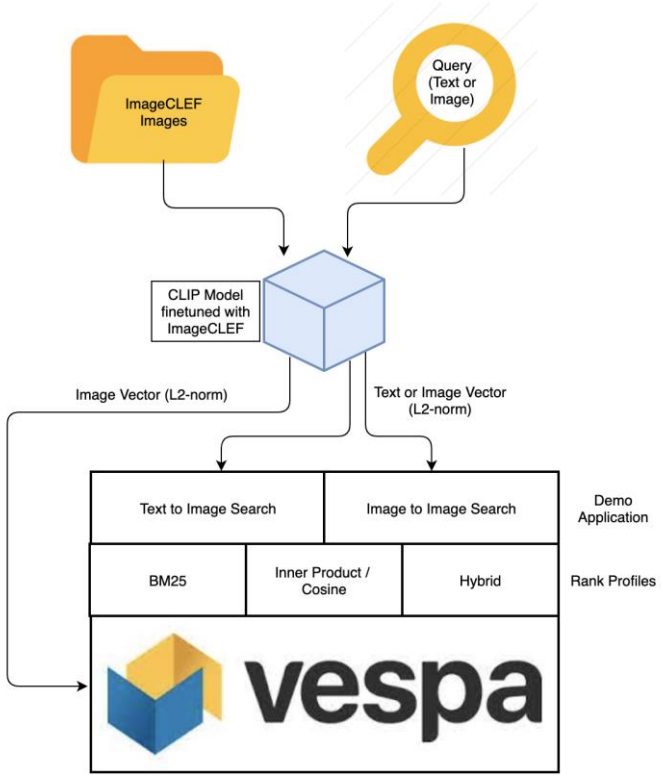


Vector Search

- **Exhaustive Search** – match the query vector to every vector in the database (O(N))
- **Approximate Nearest Neighbors** (ANN) – tradeoff accuracy for speed.
  - **Space partitioning** – see blog post by Eyal Trabelsi)
    - Using Forests (ANNOY)
    - Using LSH
    - Pre-clustering and matching centroids
    - Quantization / Product Quantization
  - **Hierarchical Navigable Small Worlds Graph (HNSW)** – hierarchy of probability skip lists (see blog post by James Briggs)

Demo Application

- We chose Vespa as our vector store
- Hybrid Search platform supports both text (BM25) and vector (HNSW) search, allowing easy comparison of legacy and new behavior
- Our demo application allows:
  - Text to image search – using caption text, caption vector, and hybrid
  - Image to image search – using image vector, and image vector + caption



Vespa Configuration

- **Schema (s/m/a/schemas/image.sd)**
  - **image_id** (string, properties: summary, attribute)
  - **image_path** (string, properties: summary, attribute)
  - **caption_text** (string, properties: summary, index)
  - **clip_vector** (tensor<float>[512], properties: attribute, index, HNSW)

```
schema image {
    document image {
        field image_id type string {
            indexing: summary | attribute
        }
        field image_path type string {
            indexing: summary | attribute
        }
        field caption_text type string {
            indexing: summary | index
            index: enable-bm25
        }
        field clip_vector type tensor<float>(x[512]) {
            indexing: attribute | index
            attribute {
                distance-metric: innerproduct
            }
            index {
                hnsw {
                    max-links-per-node: 32
                    neighbors-to-explore-at-insert: 500
                }
            }
        }
    }
```

Vespa Configuration (cont'd)

- Ranking Profiles
  - **caption-search** – bm25(caption_text)
  - **image-search** – closeness(field, clip_vector)
  - **combined-search** – bm25(caption_text) + closeness(clip_vector, query_vector)
- Query Vector template needs to be defined in s/m/a/search/query-profiles/types/root.xml
- Managing vespa – I used homegrown scripts from (sujitpal/vespa-poc) but probably better to use the official scripts from vespa-cli.

```
    fieldset default {
        fields: image_id, image_path, caption_text, clip_vector
    }

    rank-profile caption-search inherits default {
        first-phase {
            expression: bm25(caption_text)
        }
    }

    rank-profile image-search inherits default {
        first-phase {
            expression: closeness(field, clip_vector)
        }
    }

    rank-profile combined-search inherits default {
        first-phase {
            expression: bm25(caption_text) + closeness(clip_vector)
        }
    }
}
```

Demos

Demo Links

- **Demo link for clip-imageclef**
  - internal, need to be in Elsevier VPN
  - http://10.169.23.142/clip-demo
- **Demo link for clip-rsicd**
  - publicly available on HuggingFace spaces
  - https://huggingface.co/spaces/sujitpal/clip-rsicd-demo

Resources

- Code for fine-tuning CLIP with ImageCLEF data and image search using Vespa – [elsevierlabs-os/clip-image-search]

Thank you

sujit.pal@elsevier.com
https://twitter.com/palsujit
https://www.linkedin.com/in/sujitpal/