Embedding vector-based multimodal search

Sairisheek Muttukuru
University of California, Los Angeles
Los Angeles, CA 90024
sairisheekm@gmail.com

Shiyao Guo
University of California, Los Angeles
Los Angeles, CA 90024
sguo18@g.ucla.edu

Lily Takahari
University of California, Los Angeles
Los Angeles, CA 90024
lilytakahari@g.ucla.edu

Rohit Karmarkar
University of California, Los Angeles
Los Angeles, CA 90024
rohitsk@g.ucla.edu

Abstract

In this report, we discuss efforts to embed text and images into a global feature space. We also highlight the databases that efficiently search the resulting embedding vectors. We present a search-engine like product which composes two publicly available tools, CLIP and Milvus, which respectively accomplish the previous tasks, and evaluate their performance on a image-caption dataset depicting everyday scenes and objects. We conclude that the ideas these tools are based on have promising applications in a variety of downstream tasks.

1 Introduction

In the modern era, copious amounts of data are collected from myriads of sources like mobile phones, smart watches, satellites etc. Big data spans different types and formats e.g. text, images, audio, HTML etc. By jointly embedding related multimodal data into a global feature space, we can model, query, and search the data in a uniform manner. Additionally, we can use the embeddings for downstream machine learning (ML) tasks like zero-shot classification and regression. We also want to be able to search embeddings for large-scale data efficiently.

Open-source ML models like OpenAI's Contrastive Language-Image Pre-training network (CLIP) jointly embed text and images to minimize the cosine similarity between the embeddings for related images [3]. Vector search databases and tools like Milvus [4] and FAISS [2] build search indices for embedding vectors on a trillion scale. For this project, we explore the composition of CLIP and Milvus to build a search engine-like product that supports search of an uploaded image or text and finds similar images or texts (captions) for existing images in a dataset. The dataset we use is MS COCO (Microsoft Common Objects in Context).

2 Motivating/Referenced Work: MS COCO, CLIP, Milvus

Microsoft's COCO dataset contains images of everyday objects, people, and scenes. Among other annotations for computer vision purposes, it contains five captions per image. We download their train2017 data split containing around 110k images and use their COCO API to load captions [1].

OpenAI released their CLIP models in 2021. Their intent was to learn transferable visual models from natural language supervision, meaning that their model would be able to handle different downstream tasks without training specifically on those tasks. For example, CLIP can handle zero-shot classification, e.g. matching a text description to an image without having been trained on similar

Preprint. Under review.

images. Their approach was to learn a “multi-modal embedding space by jointly training an image encoder and text encoder to maximize the cosine similarity" between the embeddings of matching (text, image) pairs while minimizing the similarity between non-matching pairs [3]. They trained several models with different components and parameters on their dataset of 400 million (text, image) pairs sourced from the Internet. Their best performing model is ViT-L/14@336px, which outputs embeddings of dimension 768 [3]. We use this model and its API to obtain embeddings of the COCO images and their captions.

Milvus is a vector database created by the Linux Foundation AI and Data Foundation that was initially released in 2019. It is intended to “store, index, and manage massive embedding vectors generated by deep neural networks and other machine learning (ML) models" [4]. It supports various index types depending on the user’s search requirements like speed vs. accuracy. Since our dataset is not quite million-scale, we can use the FLAT index, which returns “perfectly accurate and exact search results on a small, million-scale dataset" [4]. We use Milvus Standalone, which is a Docker container that we connect to through a local port.

3 Methods

We created a Google Cloud Compute Engine instance on which to store COCO data, the CLIP model, and the Milvus server. To enable the search for similar images and captions in COCO, there are two stages of our workflow: offline work, during which COCO data embeddings are first stored in Milvus, and online work, when users of our product provide their search target and we return results through a graphical interface.

3.1 Offline Work

For our offline work, we downloaded the train2017 COCO images and annotation files to the instance, as well as the COCO API which loads image IDs, annotation IDs, and the annotations themselves from COCO’s annotation JSON files. We then use the pre-trained ViT-L/14@336px CLIP model to obtain embeddings for each caption and image and store the embeddings into one Milvus table (i.e. Milvus Collection). Milvus only supports integers, strings, floats, and float or binary vectors, so we use the schema denoted in Fig. 1 and store the image ID or caption ID associated with the embedding. Fig. 2 gives a visual representation of the entire process. With around 110k images and usually 5 captions for each image, we end up with around 710,000 entries in the collection.

Field Name	Field Type	Attributes	Description
"id"	INT64	auto_id=False, is_primary=True	Caption ID or Image ID
"type"	VARCHAR	max_length=20	For our dataset, the value is either "caption" or "image"
"embedding"	FLOAT_VECTOR	size=768	Embedding obtained from CLIP

Figure 1: Schema for the Collection containing COCO data.

3.2 Online Work

For the online work, we created a website for the graphical interface into our system. We used Flask, a micro web framework written in Python, to create the website. On the homepage of our website, users can upload an image of any kind and click submit. Once submitted, the website will return the top 10 captions in COCO that describe the image and the top 10 most similar COCO images to the user. A separate page also allows users to type any text, and the website similarly returns the top 10 images and captions. To obtain these results, our code must obtain the CLIP embedding of the user’s input in order to use Milvus’ k-Nearest Neighbors (k-NN) search. We use L2 distance as the search metric because the embeddings are not normalized. From Milvus we obtain the image or caption

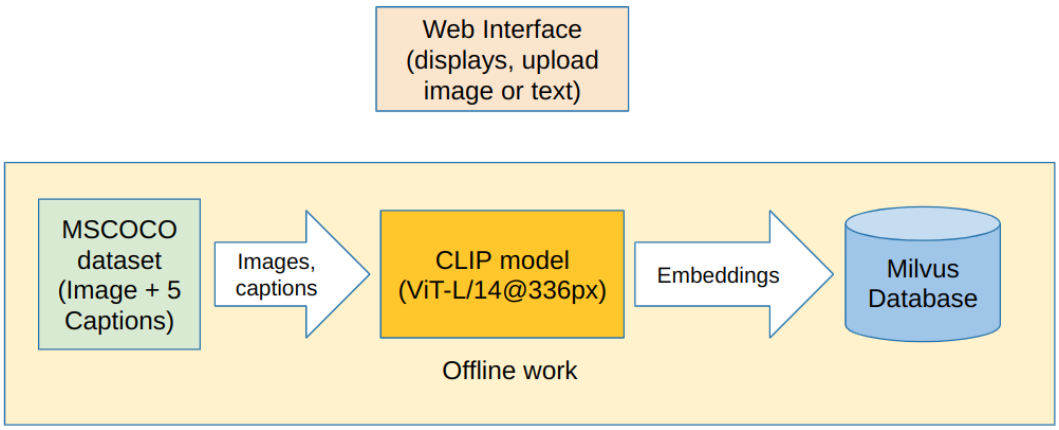


Figure 2: Offline work: Storing COCO data onto Milvus for querying later.

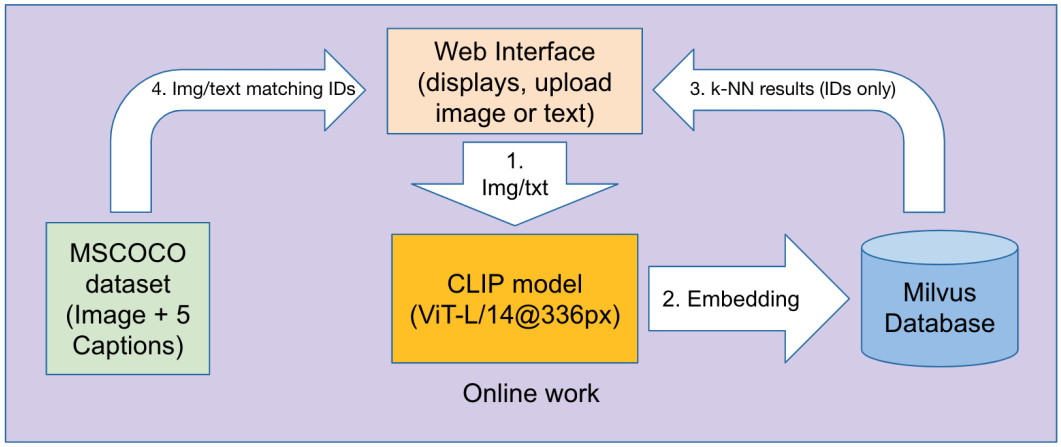


Figure 3: Online work: Querying and search in the stored data on Milvus using the interface.

IDs, so we use the COCO API to load the appropriate content. The content is then displayed on the interface. Fig. 3 demonstrates the steps of this process.

4 Results and Evaluation

We include screenshots of the interface in the Appendix, Fig. 7. As expected, providing an existing COCO image or caption returns the same item as the most similar image or caption result, respectively. This demonstrates that Milvus will accurately return the closest vector to the given vector. However, if we were to use a different index than FLAT, the results would have lower precision and recall. The other vector indices trade exact k-NN results for approximate k-NN results on much larger, billion-scale sets of data, while still maintaining speed [4]. Thus, instead of evaluating the precision and recall of Milvus on embeddings, we were more interested in evaluating the speed of our system and whether CLIP's model was able to embed text and images into a global feature space such that we obtain related results when the search crosses data modes. Specifically, we evaluate image->caption search and visualize CLIP embeddings with principal component analysis.

4.1 Effectiveness and Speed of Image to Caption Search

We can qualitatively evaluate the results given by the image->caption search. In the Appendix's Fig. 7, the returned captions mentioning surfers seem unrelated to the given image showing a man giving

a presentation. However, an image like a horse on a beach does return captions mentioning horses and beaches. Future work can examine why CLIP embeds the caption and images into seemingly erroneous locations in the global feature space.

We quantitatively evaluate our search by randomly choosing 5000 image IDs out of the 110k COCO images and seeing if the returned captions match the image's 'true captions' i.e. if they are associated to the image in COCO. First, we attempt to measure precision@k and recall@k based on whether the k-NN search returns the exact true captions. Thus, precision@k = (# true caption IDs in the k returned search IDs)/(# total search caption IDs = k) and recall@k = (# true caption IDs in the k returned search IDs)/(# total true caption IDs, which is usually 5). We evaluate this at different k's as shown in Table 1; naturally, precision decreases as k increases and more search results are returned, and recall increases as more true captions could be found in the k results. The recall@k results show that sometimes one true caption can appear in the search captions, but it is not always the case since the average is less than $1/5$. This is reasonable because CLIP is not expected to embed COCO images and captions in a way that the exact true caption will be matched up with the image.

Since the results of precision and recall are not particularly satisfying, we also convert all true captions into a collective bag-of-words and all k search captions into a collective bag-of-words. We then compute the Jaccard and cosine similarity between the two. Jaccard is lower than cosine since it ignores word counts and looks at unique words only. Cosine similarity gives higher values because it considers the counts of the unique words. These two metrics also decrease and increase w.r.t. an increasing k because of the additional words added on by the additional returned search captions. Cosine similarity increasing with additional k could indicate that the additional captions still share the same keywords/topics, so CLIP appears to at least be able to embed on the granularity of a general topic related to the image and captions.

Finally, we measure the time taken to return the results for the 5000 images. It only took 3.54 seconds to retrieve the 5000 embeddings, querying by ID. Table 2 shows that it took around 30 ms per image to search for the embeddings of the top k captions. If we had a billion-scale dataset, it would likely take much longer, at which point we would need to consider Milvus' other vector index options.

k	Precision@k	Recall@k	Jaccard	Cosine
5	0.1161	0.1161	0.3008	0.6127
7	0.0954	0.1335	0.2888	0.6341
10	0.0770	0.1540	0.2684	0.6529

Table 1: Evaluating CLIP's effectiveness at embedding an image and related captions closer together in the global feature space. Metrics are averaged across 5000 images.

k	Total time (s)	Per result (s)	Per query (s)
5	147.456	0.00589	0.0294
7	153.912	0.00439	0.0307
10	156.554	0.00313	0.0313

Table 2: Time taken for image->text search for 5000 images.

4.2 T-SNE Analysis of CLIP Embeddings

We applied a dimensionality reduction method to the CLIP embeddings to visualize the distances between images and captions separately. We have included screenshots in the Appendix. The intent is to determine whether CLIP still preserved pairwise distances between images or the captions (i.e. the joint embedding scheme did not sacrifice similarity between the same mode). T-SNE was applied instead of a linear dimensionality reduction method such as PCA because we are more interested in analyzing small pairwise distances versus the preservation of large pairwise distances (i.e. we care more about preserving similarities than dissimilarities). The T-SNE analysis of the captions showed that the joint embedding preserved similarities between captions extremely well. Multiple perplexity values were tried and the value with the most apparent clusters was chosen. The algorithm was also iterated until a point of stability.

For T-SNE performed on the images, we observed that the quality of the clusters was not as cohesive as the captions. Possible reasons are that the feature space of the images is much larger and more

complex than that of the captions, or that the multimodal embedding deteriorated the pairwise similarity of the images slightly.

5 Limitations and Future Work

As mentioned earlier, we could potentially gain insight into better ways to embed images or text if we investigate why we encounter nonsensical results. We could have evaluated Jaccard and cosine similarity for all COCO images instead of just 5000, but we were limited by time and computation resources. Additionally, we could use different evaluation metrics for the image->caption search, or create appropriate metrics to evaluate text->image search.

If we had additional time and computation resources, we could have further trained CLIP's pre-trained models on MS COCO (caption, image) pairs. We would then expect that the effectiveness of the image->caption search would improve. CLIP as an overall technique can be used to train joint text-image embedding models. For example, we could create models for the MIMIC chest X-ray database to find the diagnoses that were associated with X-rays similar to a given query X-ray. This is a more useful real-world application of CLIP and Milvus than our project. Another application could be embedding online shopping searches and their results as a way to improve recommendation systems.

6 Conclusions

As ML advances, and as more ML applications tie together multimodal data, the idea of embedding data into a global feature space becomes more promising. Designing databases based on this idea additionally makes the idea cost-effective. Our project demonstrates just one application that takes advantage of this idea, with many more applications possible in the field of medicine, e-commerce, and more.

References

[1] *COCO - Common Objects in Context*. https://cocodataset.org/#download. Accessed: 2022-12-12.

[2] *Facebook AI Similarity Search*. https://faiss.ai/. Accessed: 2022-12-12.

[3] Alec Radford et al. "Learning Transferable Visual Models From Natural Language Supervision". In: *CoRR* abs/2103.00020 (2021). arXiv: 2103.00020. URL: https://arxiv.org/abs/2103.00020.

[4] *Vector database - Milvus*. https://milvus.io/. Accessed: 2022-12-12.

Appendix

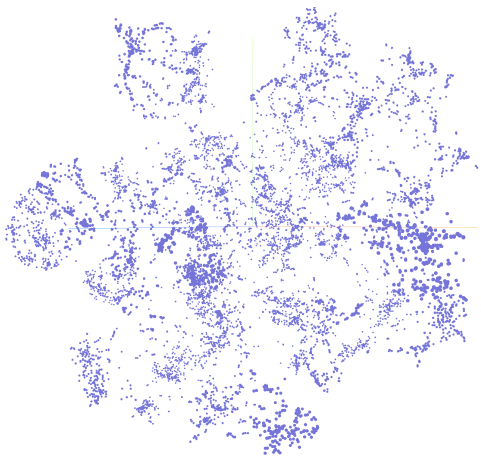


Figure 4: T-SNE plot of captions.

Figure 5: The embeddings semantically related to tennis are paired closely together.

Figure 6: T-SNE plot of images.

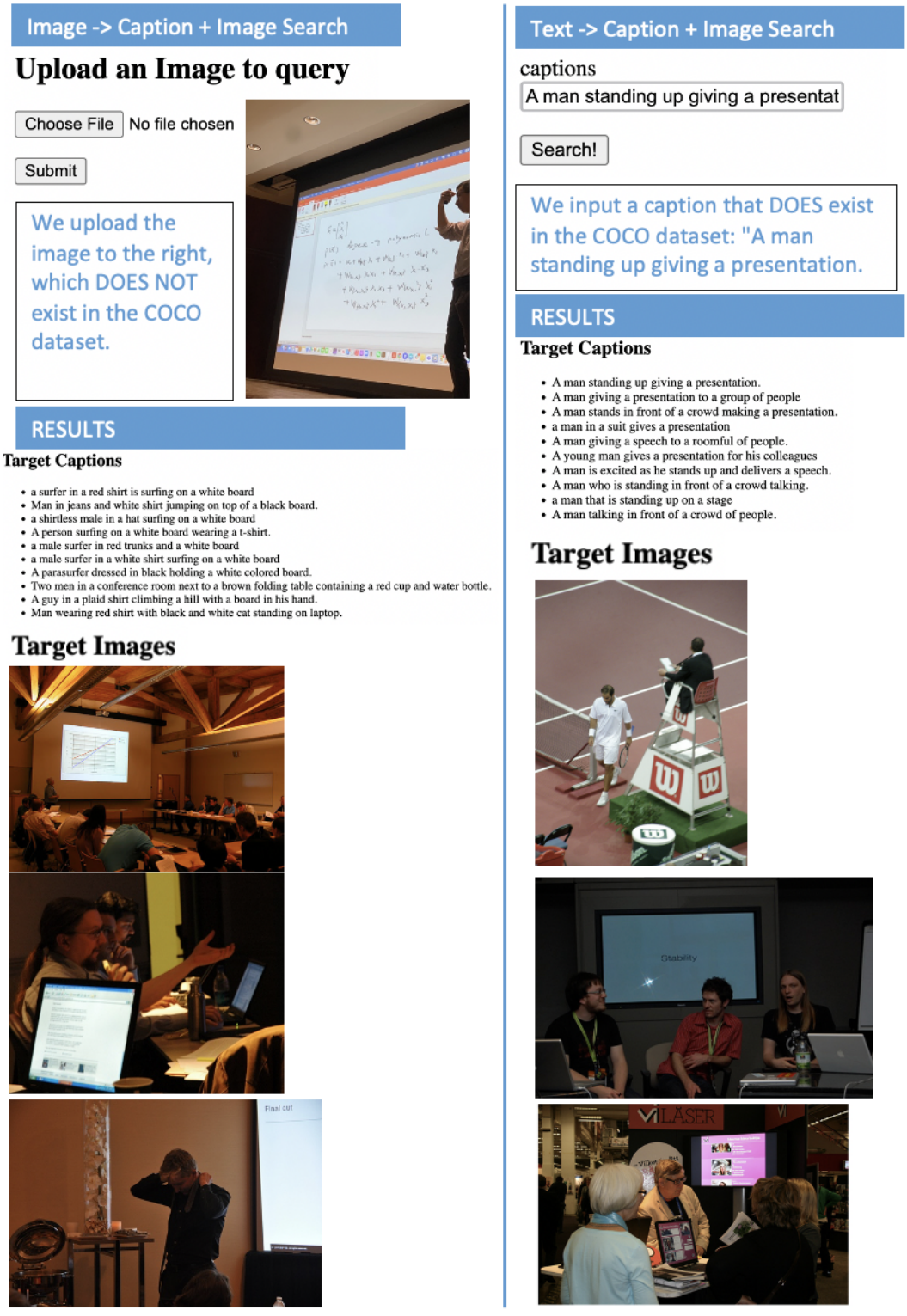


Figure 7: Screenshots of our interface and the results of two example queries. The example image does not exist in COCO while the example text does. Thus, the example text appears as the top result in the returned captions. Additionally, only the top 3 images are shown due to space constraint; the website displays all top 10 images.

UCLA Samueli School of Engineering

Embedding vector-based multimodal search

Lily Takahari (305108348)
Sairisheek Muttukuru (905433008)
Shiyao Guo (805949408)
Rohit Karmarkar (505848825)

Outline

1. Motivation and current work
2. Block diagrams of our process
3. Results
4. Evaluation
5. Live Demo and Q&A
6. (not presented on Thursday) Conclusions and Future Work

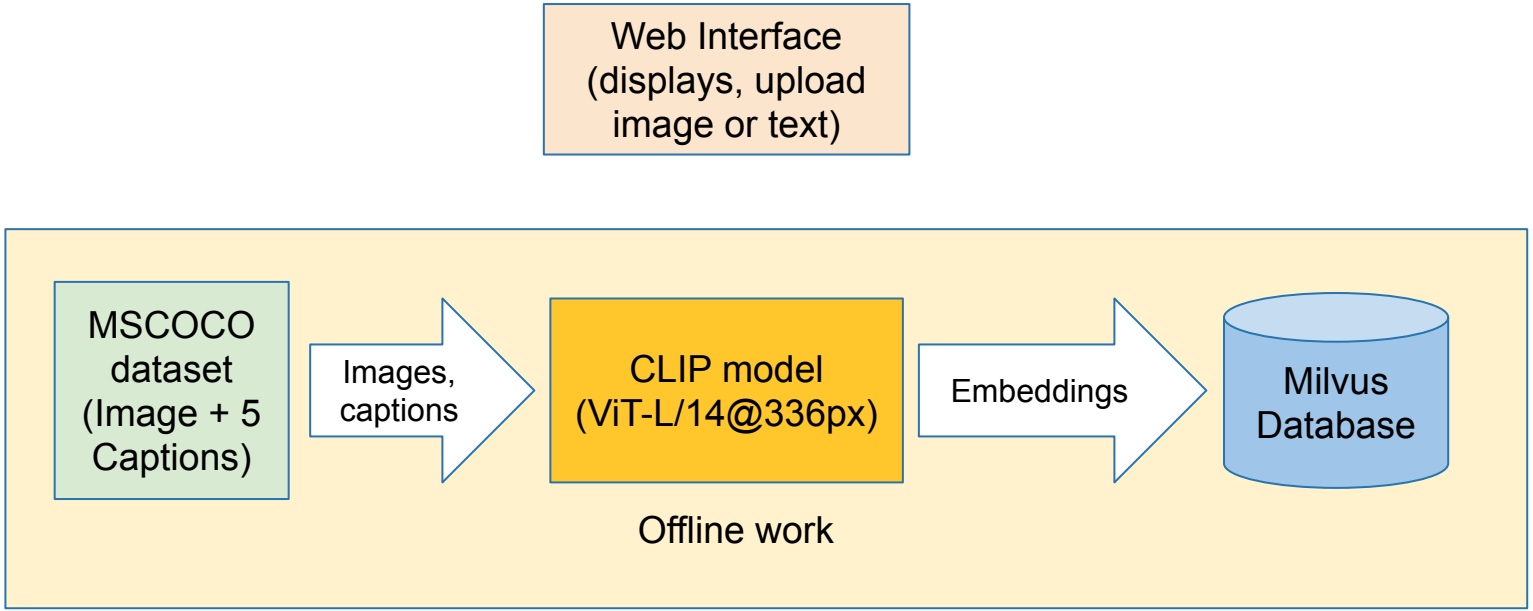
Motivation and current work

- We want to jointly embed data such that related multimodal data have similar embeddings
  - **Motivation:** Embeddings can be used for downstream ML tasks like classification
  - **Current work**: CLIP techniques for text+images
  - For the captions of an image, the image and captions should have similar embeddings i.e. low L2 distance
- We want to search for similar embeddings among large-scale data efficiently
  - **Motivation:** ML tasks, as well as big data analysis tasks
  - **Current work:** Milvus, Faiss

- **Goal:** Compose these publicly available tools to build a search engine-like product that supports search of an uploaded image or text for similar images or captions in a database
  - Use a Google Compute Engine Virtual Machine

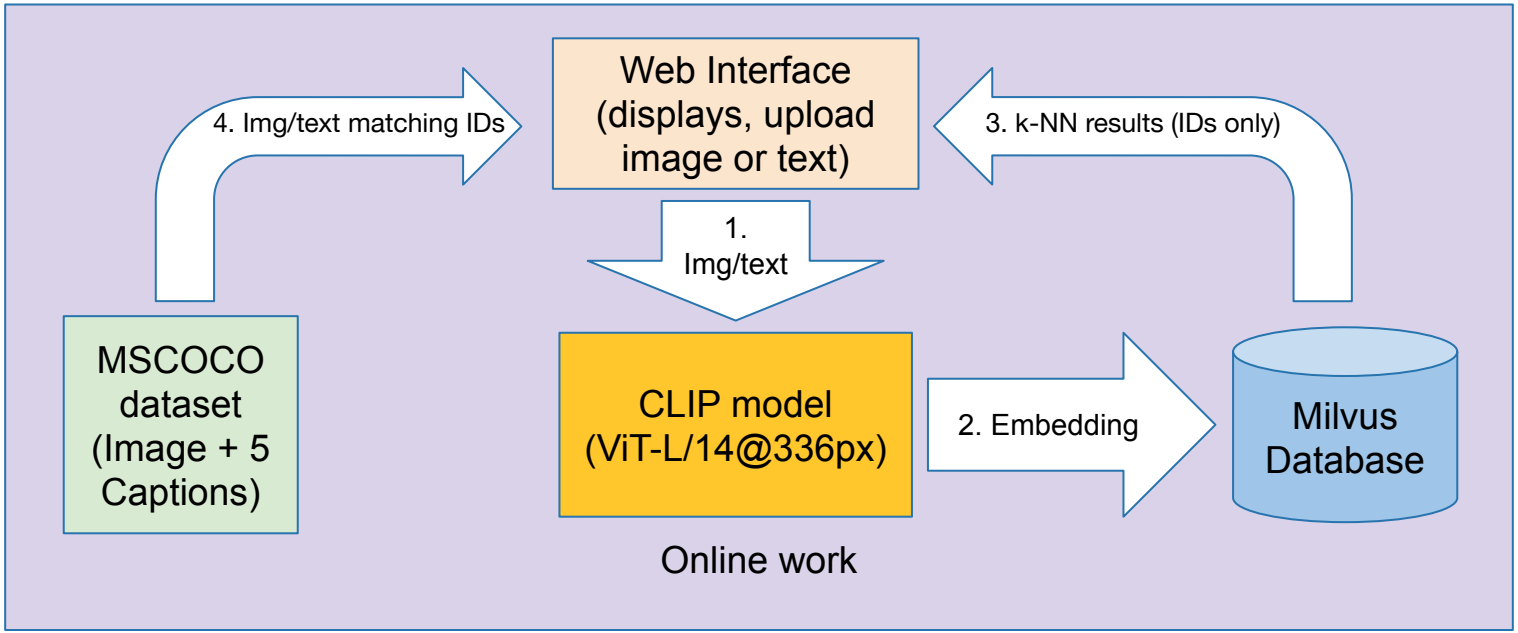
Storing embeddings

Offline work: Storing data onto Milvus for querying later

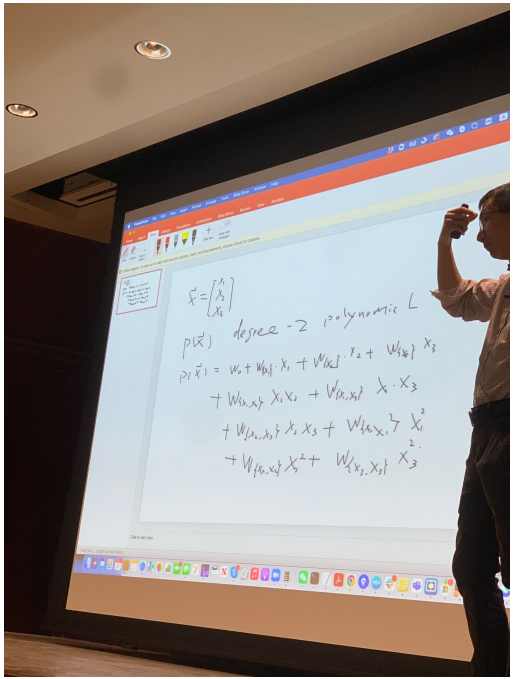


Searching embeddings

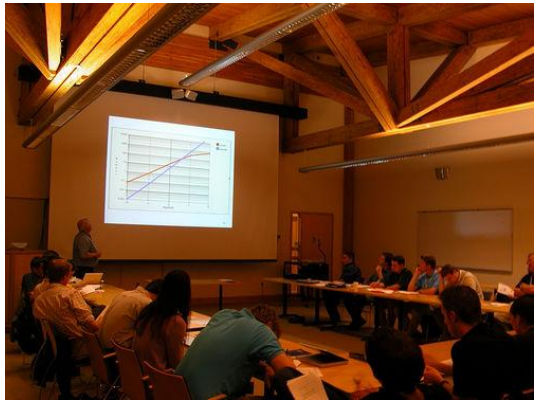
Online work: Querying and search in the stored data on Milvus with the interface



Results



Top 3 images in MSCOCO train2017 split

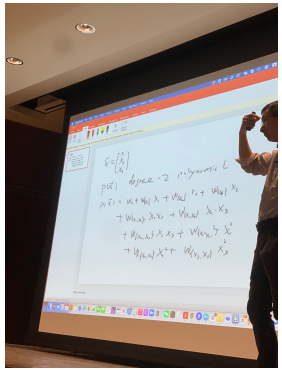


1. L2 Distance: 253.689

2. L2 Distance: 259.472

3. L2 Distance: 262.276

Results

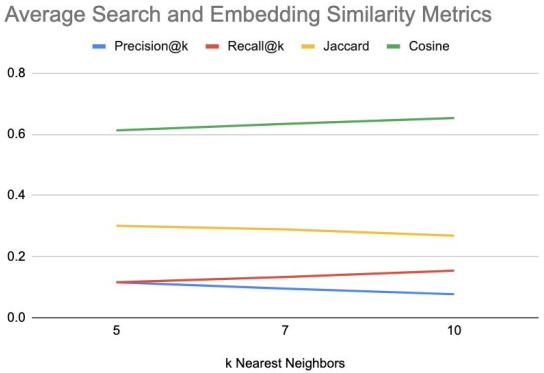


Top 10 captions

1. 345.974; a surfer in a red shirt is surfing on a white board
2. 346.431; Man in jeans and white shirt jumping on top of a black board.
3. 347.668; a shirtless male in a hat surfing on a white board
4. 347.845; A person surfing on a white board wearing a t-shirt.
5. 348.001; a male surfer in red trunks and a white board
6. 348.461; a male surfer in a white shirt surfing on a white board
7. 349.363; A parasurfer dressed in black holding a white colored board.
8. 349.595; Two men in a conference room next to a brown folding table containing a red cup and water bottle.
9. 349.965; A guy in a plaid shirt climbing a hill with a board in his hand.
10. 349.998; Man wearing red shirt with black and white cat standing on laptop.

Evaluation: k-NN and runtime

- Process: Randomly choose 5000 embeddings from around 110k embs on Milvus, search k-NN captions at different k (image -> text search)
  - Query time for getting 5000 embeddings by image id: 3.54 seconds
- Trends in the first graph occur because of increase in k
  - Also, reflects more about CLIP's ability to embed similar data, rather than Milvus' search accuracy
  - Converted all 5 true captions into combined Bag-of-Words, all k search result captions into BoW, to compute Jaccard and cosine similarity



k	Precision@k	Recall@k	Jaccard	Cosine
5	0.1161	0.1161	0.3008	0.6127
7	0.0954	0.1335	0.2888	0.6341
10	0.077	0.154	0.2684	0.6529

k	Seconds	Seconds per result	Seconds per search target
5	147.456	0.00589824	0.0294912
7	153.912	0.00439748571	0.0307824
10	156.554	0.00313108	0.0313108

Evaluation: CLIP Embedding Visualization

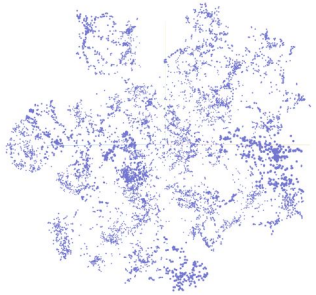


Figure 4: T-SNE plot of captions.

Figure 6: T-SNE plot of images.

Figure 5: The embeddings semantically related to tennis are paired closely together.

Live Demo and Q&A

Fun captions that exist in the MSCOCO dataset:

- I do not know what this image is.
- This is image does not have photo attached to it so not sure what to write here
- People are working on something that is interesting.
    - That’s all of us in CS 260. Thank you!

Conclusions and Future Work

- Pre-trained CLIP works decently for general images and captions
  - CLIP as an overall technique can be used to train joint text-image embedding models e.g. train on the MIMIC chest X-ray database to find similar diagnoses for a given X-ray
  - Embeddings of similar images are more strongly similar than image with their captions, which is expected, because the image-text relationship is the difficult problem that CLIP needs to learn
- Search on multimodal embeddings can be generalized beyond text-image, just need to train models to give those embeddings