Unlocking the Potential of Unlabelled Data with Zero Shot Models & Vector DBs | by The Tenyks Blogger | Medium

Key Topics	- Unlocking the potential of unlabelled data - Zero shot models - Vector DBs
Summary	This article explores the potential of using zero-shot models and vector databases to leverage unlabelled data. It discusses how these approaches can unlock new possibilities in data analysis and machine learning tasks.
Colabs Notebook	https://medium.com/@tenyks_blogger/unlocking-the-potential-of-unlabelled-data-with-zero-shot-models-vector-dbs-4a64d5e61c6d
Created time	@April 25, 2024 9:56 PM

Unlocking the Potential of Unlabelled Data with Zero Shot Models & Vector DBs

The Tenyks Blogger
The Tenyks Blogger
.
Follow
10 min read
.
Sep 4, 2023
1

Unlock Insights from Unlabelled Images in Machine Learning: Exploring Object Detection Datasets with and without Labels.

Search and retrieval of thousands or even millions of images is a challenging task, especially when the data lacks explicit labels. Traditional search techniques rely on keywords, tags, or other annotations to match queries with relevant items. But what if the data is unlabelled?

In this article, we'll explore some cutting-edge machine learning techniques that can enable search over unlabelled data. Using a combination of vector embeddings, vector databases and zero-shot models, we can build powerful search systems over unlabelled data in computer vision.

Table of Contents

1. Fundamentals

   Embeddings

   Embeddings are a machine learning technique for representing pieces of data as vectors in a continuous vector space. The key idea is that the position and orientation of the vectors captures semantic properties of the data. Vectors that are closer together in the space are more similar, even if the data itself is unlabeled.

   Embeddings power many modern AI systems because they provide a simple yet scalable way to represent complex data in a format that algorithms and models can utilize. They can uncover hidden relationships and patterns that were not obvious before, enabling new ways to search, organize, and analyze data without the need for manual labelling.

   Multi-modal embeddings

   Multi-modal embeddings are embeddings that represent data from multiple different modalities in a single vector space.

   📣 For more information on this topic, see our article on Multi-Modal Search.

   The key idea is that the embedding space is aligned so that images, text, sounds, etc. that are semantically related will have embedding vectors that are close together, even though they are from different modalities.

   OpenAI developed CLIP [1], a model capable of acquiring shared representations of images and their corresponding textual descriptions. Through training on extensive datasets consisting of images and their associated captions, CLIP is proficient at embedding images and texts into

a latent space that they both share.
By combining multiple modalities in a shared embedding space, multi-modal embeddings provide a more powerful way to represent complex data and relationships. They enable a deeper kind of understanding and more flexible knowledge-based reasoning than using any single modality alone.
Vector DBs
A vector database is a specialized type of database specifically designed to store and manage large amounts of embeddings efficiently. It is optimized for similarity searching tasks, making it well-suited for applications such as image search, recommender systems, and various other domains.
The vector database employs advanced indexing structures and algorithms specifically tailored for fast similarity search. These indexing techniques organize the embeddings in a way that optimizes the search process, reducing the computational complexity and enabling fast retrieval of nearest neighbours to a given query vector.
You can learn more about Vector DBs in our previous article.
Zero-Shot models
In computer vision, Zero-Shot models [2, 3, 4] are machine learning models that can recognize and localize new object categories in images without any labelled examples of those categories.
Traditional object detectors require thousands of images with bounding boxes or instance segmentations to learn each new class. Zero-Shot object detectors can learn to detect new classes by transferring knowledge from other seen classes:
For example, if a detector has been trained to find cats, dogs, and birds, it may be able to detect giraffes as well by understanding that they are also four-legged animals with a torso, neck and head. The model learns an embedding space where animal classes that share visual and semantic features are close together, like the embeddings of “furry”, “four legs”, “tail”, and so on. Once the model has learned these associations between the shared attributes of classes, it can apply them to a new class by inferring where it may lie in the embedding space.
Zero-Shot object detection [5] opens up the possibility for detecting thousands of object categories without requiring datasets of images for each class. This could enable applications like robotics in unconstrained environments where the types of objects are extremely varied.

2. Pre-processing pipeline
   To enable search in unlabelled image datasets, we first pre-process the raw

images into a format that supports similarity-based search. Specifically, we go through several stages to transform unlabelled images into vector embeddings, which are then indexed in a specialized vector database.
By indexing these embeddings, we can find results that are highly relevant to our queries even without any class labels or bounding box annotations attached to the images.
The pre-processing pipeline consists of four main stages:
Collecting a set of raw, unlabelled images.
Run the data through a Zero-Shot detection model.
Generate the vector embeddings of the detection model outputs using an embedding model (⚡Note: CLIP is only one option of many foundational models that can be used for this step).
Index the resulting embeddings in a Vector DB to do similarity search.
In the following sections, we explore each stage of the pre-processing pipeline highlighting how vector search over embeddings can reveal relationships between unlabelled images.
2.1 Collecting a set of unlabelled images

Figure 2. In the first stage we collect unlabelled data relevant to our use-case
We start by compiling an ideally large set of raw, unlabelled images relevant to our use case (Figure 2). If our goal is to search over mechanical failures in manufacturing scenes, we may gather images of mechanical devices exhibiting corrosion or erosion failures. The images should be representative of the types of content users may search for. At this stage, no annotations or labels are applied. The only constraint is collecting enough images to enable our models to learn rich feature representations and similarity spaces for search.
2.2 Pass the data through a Zero-Shot detection model

Figure 3. The Zero-Shot model is used to generate objects with no training involved
We apply a Zero-Shot model or Foundational Models to process our unlabelled image dataset (Figure 3). The goal is to use these models to predict objects which, in addition to the unlabelled images, will be run through an embedding model to generate embeddings.
There are different options that can be used for this step, we summarize some of them here:
Segment Anything [6], or SAM, a Foundational Model with zero-shot generalization to unfamiliar objects and images, without the need for additional training. It employs a transformer-based mask decoder that predicts segmentation masks for objects from the image and prompt embeddings. The

outputs of SAM (i.e. instance segmentation) are then cast to bounding boxes in a post-processing step.
Region CLIP [7] extends CLIP to enable object detection at a region level rather than matching whole images to text. It leverages a CLIP model to match image regions to template captions, then pre-trains a model to align these region-text pairs in the feature space. This mitigates CLIP's limitation of capturing only image-level semantics rather than region-level alignment.
Grounding DINO [8] is an open-set object detector that marries the DINO detector with grounded pre-training to detect arbitrary objects specified by inputs like category names or referring expressions. It introduces language to a closed-set detector for open-set generalization.
OWL-ViT [9] uses a standard Vision Transformer, contrastive image-text pre-training, and end-to-end fine-tuning for open-vocabulary object detection. OWL-ViT requires adaptation strategies and regularization to achieve strong results on Zero-Shot text-conditioned detection and one-shot image-conditioned detection.
👨‍🍳 Secret Sauce: the above options are all open-source, but this could also be a good opportunity to explore closed-source options as well, such as fine-tuned foundational language models or neural-based retrieval methods, which is something we have experimented at Tenyks.
2.3 Extract object-level and image-level embeddings

Figure 4. An embedding model, e.g. CLIP, is used to transform objects and images into embeddings
For object-level embeddings, Figure 4, we input image regions into CLIP (or another similar foundational model) and get a vector representing their visual contents. To obtain image-level embeddings, we pass the full images into CLIP. By using a model such as CLIP as a generic feature extractor, we can get embedded representations to power similarity search over our non-annotated images— the semantics are implicitly encoded in CLIP's learned embedding space.
Since Region CLIP and OWL-ViT use CLIP during the object detection phase, hence with some tweaks we can manage to directly output the embeddings from these two models, rather than adding an extra step as in the SAM or the Grounding DINO pipeline.
2.4 Store the vector embeddings in a VectorDB

Figure 5. The vector database indexes the embeddings
Once we have our object and image embeddings from CLIP, we need to index them for fast search (Figure 5). We use a vector database, or VectorDB,

optimized for managing and querying high-dimensional embedding spaces. The VectorDB stores the embeddings for efficient nearest neighbour search so that we can find the closest matches to any new query vector. Because vector similarity corresponds to semantic similarity in CLIP's embedding space, nearest neighbour search allows us to retrieve the images most relevant to a query based on the learned visual semantics.

3. Search pipeline

Figure 6. Search pipeline in the Tenyks platform including the unlabelled data pre-processing branch

The high-level overview of the search pipeline, which includes the unlabelled dataset pre-processing branch, is depicted in the figure above. Once both the unlabelled and labelled sets are fed into the embedding model, the resulting output, i.e., the embeddings, are forwarded for indexing by the Vector DB. This crucial stage is referred to as the Indexing Stage, as illustrated in the preceding image, Figure 6.

A more detailed breakdown of Tenyk's multi-modal search pipeline is presented in our article on this topic.

4. Showcase: searching through unlabelled data

We showcase multi-modal search including unlabelled and labelled data in the Tenyks platform.

Let's consider the BDD dataset, a comprehensive driving dataset, where our objective will be to conduct image searches based on three distinct modalities. By leveraging the power of multi-modal search, we can efficiently explore and retrieve relevant images from the dataset, enhancing our understanding and analysis of the driving domain.

Text search: querying for "taxis"

To continue our exploration, we can search for images of taxis using the query "taxis". As depicted in Figure 2, upon inputting the text query, we retrieve a collection of images, each of which prominently features images with taxis. This allows us to swiftly identify and access relevant visual content pertaining to taxis within the dataset.

Figure 7. Searching for similar images based on the text "taxis"

Image search: querying for "crosswalks in nighttime conditions"

Furthermore, we can expand our search capabilities by providing a specific image as input. In Figure 8, we showcase the functionality of the search feature by selecting an image featuring crosswalks in nighttime conditions. As a result, the search engine successfully retrieves a set of images that share very similar scenes.

Figure 8. Search for similar images based on image: crosswalks in nighttime settings

Object-level search: querying for “trucks”

Finally, for a more refined search, we can leverage objects instead of selecting whole images. In Figure 10, by intentionally choosing a “truck” object, the search engine retrieves similar objects that closely resemble the selected one. This capability enables precise exploration and identification of specific object categories within the dataset.

Figure 9. Searching for similar images at the object-level: “trucks”

5. Summary

This article provides an overview of how to search through unlabelled data by leveraging Zero-Shot models, embedding models and Vector DBs.

The first section explains the basics, including the role of embeddings in representing data, the importance of vector databases for efficient indexing, and the capabilities of zero-shot models.

The second part delves into the pre-processing pipeline, outlining the four key stages involved in transforming unlabelled data into vector embeddings.

Moving forward, the article explores the big picture of the search pipeline in the Tenyks platform, highlighting how the embedded data (corresponding to both unlabelled and labelled data) is indexed and retrieved.

Lastly, a showcase example demonstrates the power of the framework in searching through unlabelled data. By leveraging the pre-processed embeddings and search pipeline, the Tenyks platform enables users to perform targeted searches, even with minimal or no labels. This capability opens up new possibilities for discovering valuable insights and patterns within unlabelled datasets.

References

[1] Learning Transferable Visual Models From Natural Language Supervision

[2] An embarrassingly simple approach to zero-shot learning

[3] Zero-Shot Model Diagnosis

[4] Zero-Shot Learning Through Cross-Modal Transfer

[5] Zero-Shot Object Detection

[6] Segment Anything

[7] RegionCLIP: Region-based Language-Image Pretraining

[8] Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection

[9] Simple Open-Vocabulary Object Detection with Vision Transformers