Week 1 Report

Thomas Deatherage
23 August 2024

Abstracts & Summaries

1) Vector search with small radiuses

Conference/Journal: Not stated
Abstract: We introduce Florence-2 a novel vision foundation model with a unified prompt-based representation for various computer vision and vision-language tasks. While existing large vision models excel in transfer learning they struggle to perform diverse tasks with simple instructions a capability that implies handling the complexity of various spatial hierarchy and semantic granularity. Florence-2 was designed to take text-prompt as task instructions and generate desirable results in text forms whether it be captioning object detection grounding or segmentation. This multi-task learning setup demands large-scale high-quality annotated data. To this end we co-developed FLD-5B that consists of 5.4 billion comprehensive visual annotations on 126 million images using an iterative strategy of automated image annotation and model refinement. We adopted a sequence-to-sequence structure to train Florence-2 to perform versatile and comprehensive vision tasks. Extensive evaluations on numerous tasks demonstrated Florence-2 to be a strong vision foundation model contender with unprecedented zero-shot and fine-tuning capabilities.
Summary: Florence-2 is a new advanced model for computer vision tasks that can understand text instructions and generate accurate results in various forms like captions, object detection, and segmentation. It has shown strong performance in different tasks without needing prior training.
Link:
https://openaccess.thecvf.com/content/CVPR2024/html/Xiao_Florence-2_Advancing_a_Unified_Representation_for_a_Variety_of_Vision_CVPR_2024_paper.html

Scripts and Code Blocks

We're pulling in a lot more collaborators this semester. And one of the first things we're probably (there's a meeting next week to discuss this) going to do is deploy an MVP version of the application to an internet-accessible host. In keeping with that general goal, this week I tried to prepare the codebase for deployment. I don't know the specifics of what our hosting provider (likely computing resources from UoF) looks like, nor what an CI/CD pipeline might look like. Consequently, I focused on broadly-applicable improvements to the code base that will 1) improved code quality and 2) make it easier to on board the newest collaborators.

Specifically, I:
1) Added and fixed a bunch of type hints in the ingestor scripts, specifically the vector_embedder, idigbio ingestor and postgres output.
2) We already have retry logic on the queue jobs themselves. I also added retry logic to external network calls to improve robustness and performance.
3) Improved error handling and visibility.
4) Calmed down the previously very noisy vector embedder logs.
5) Used black-formatter to format a few files.

These code changes can be found here:
https://github.com/Human-Augment-Analytics/NFHM/pull/28/files

There are too many code changes to repeat here, so I'll focus on the highlights. Here's one such example detailing most of everything from above (type hinting, less noisy logs, consistent formatting with

black-formatter, etc) in the vector embedder code:

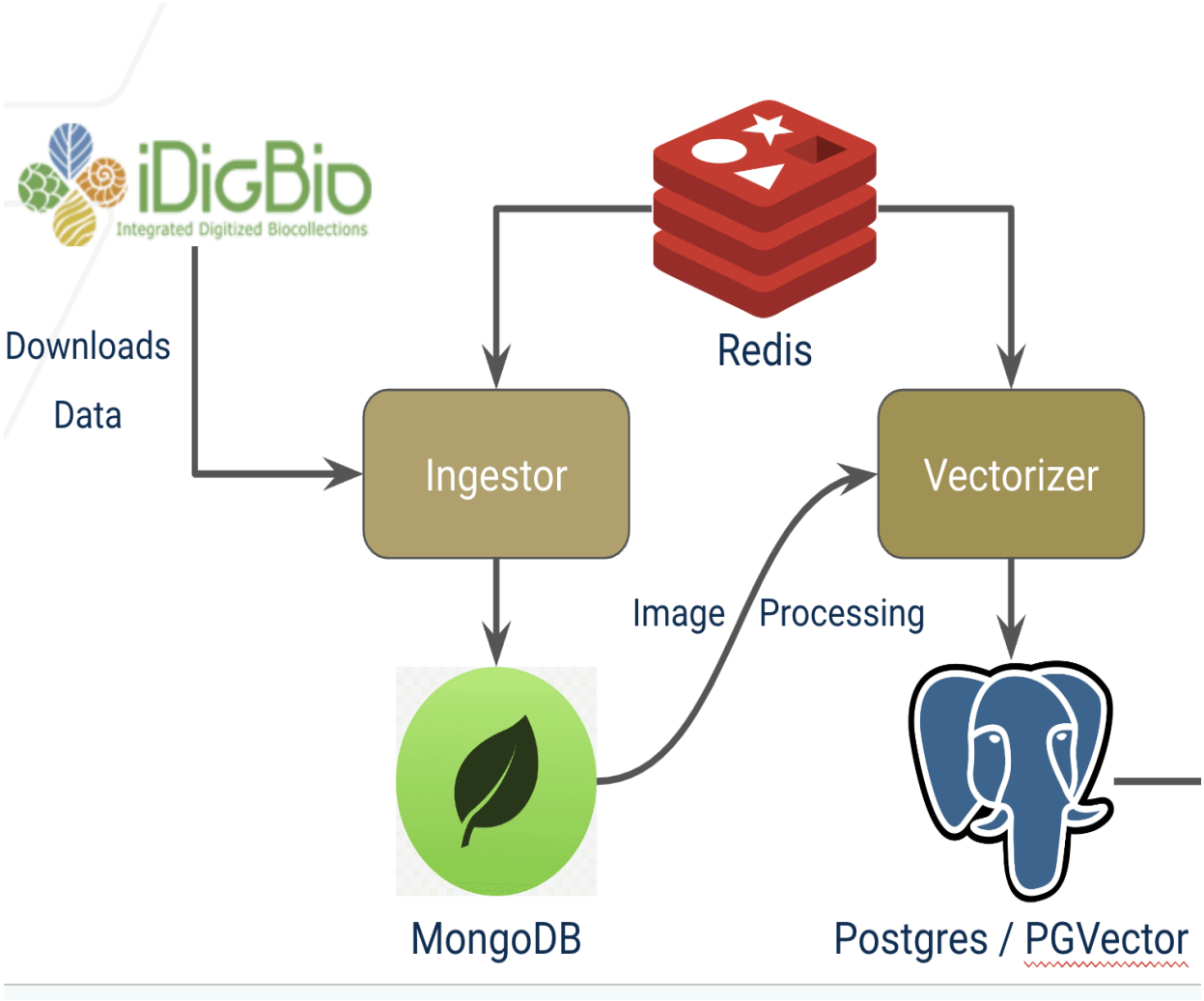
```
 import open_clip
- from datetime import datetime
- from bson.objectid import ObjectId


- logger = getLogger('vector_embedder')


  model = None
  preprocess = None
  tokenizer = None
- device = None


- if torch.backends.mps.is_available():
-     device = torch.device("mps:0")
-     logger.info("Using MPS")
- else:
-     device = torch.device("cpu")

 async def load_model():
-     """
-     Loads the pre-trained model and tokenizer for vector embeddings.
-     This is lazily evaluated once when the first call to vector_embedder is made.
-     Loading the model is a slow operation that breaks ingestor.py when done at the top level upon import
-     """
-     global model, preprocess, tokenizer, device
-     # if model is not None:
-     model, _, preprocess = open_clip.create_model_and_transforms('ViT-B-32', pretrained='laion2b_s34b_b79k', device=device)
-     model.to(device)
-     model.eval()  # model in train mode by default, impacts some models with BatchNorm or stochastic depth active
-     tokenizer = open_clip.get_tokenizer('ViT-B-32')
-     logger.info("Model loaded")
-     # else:
-         # logger.info("Model loaded already")


- def input_data_mapper(media: dict, line: dict) -> dict:
-     tmp_d = {
-         "specimen_uuid": line.get("uuid"),
-         "scientific_name": line.get("data", {}).get("dwc:scientificName") or line.get("data", {}).get("dwc:genus"),
-         "external_media_uri": media.get("data", {}).get("ac:accessURI") or media.get("indexTerms", {}).get("accessuri"),
          "media_uuid": media.get("uuid"),
          "catalog_number": line.get("data", {}).get("dwc:catalogNumber"),
          "recorded_by": line.get("data", {}).get("dwc:recordedBy"),
@@ -61,30 +62,47 @@ def input_data_mapper(media: dict, line: dict) -> dict:
          "tax_genus": line.get("data", {}).get("dwc:genus"),
          "common_name": line.get("indexTerms", {}).get("commonname"),
          "higher_taxon": line.get("indexTerms", {}).get("highertaxon"),
-         "earliest_epoch_or_lowest_series": line.get("indexTerms", {}).get("earliestepochorlowestseries") or line.get("data", {}).get("dwc:earliestEpochOrLowestSeries"),
-         "earliest_age_or_lowest_stage": line.get("indexTerms", {}).get("earliestageorloweststage") or line.get("data", {}).get("dwc:dwc:earliestAgeOrLowestStage"),
-         "collection_date": line.get("indexTerms", {}).get("datecollected")
      }

-     if line.get("indexTerms", {}).get("geopoint", {}).get("lon") is not None:
-         lat = str(line.get("indexTerms", {}).get("geopoint", {}).get("lat"))
-         lon = str(line.get("indexTerms", {}).get("geopoint", {}).get("lon"))
-         tmp_d["location"] = f"POINT({lon} {lat})"

-     if tmp_d["collection_date"] is not None:
-         tmp_d["collection_date"] = datetime.strptime('2012-01-12', '%Y-%m-%d');
+         "earliest_epoch_or_lowest_series": line.get("indexTerms", {}).get(
+             "earliestepochorlowestseries"
+         )
+         or line.get("data", {}).get("dwc:earliestEpochOrLowestSeries"),
+         "earliest_age_or_lowest_stage": line.get("indexTerms", {}).get(
+             "earliestageorloweststage"
+         )
+         or line.get("data", {}).get("dwc:dwc:earliestAgeOrLowestStage"),
+         "collection_date": None,
+         "location": None,
+         "embedding": None,
+         "tensor_embedding": None,
+     }

+     geopoint = line.get("indexTerms", {}).get("geopoint", {})
+     if geopoint.get("lon") is not None and geopoint.get("lat") is not None:
+         record["location"] = f"POINT({geopoint['lon']} {geopoint['lat']})"


+     collection_date = line.get("indexTerms", {}).get("datecollected")
+     if collection_date is not None:
+         try:
+             parsed_date = date_parser.parse(collection_date)
+             record["collection_date"] = parsed_date.date()
+         except ValueError:
```

Here’s another code sample from the idigbio ingestor code:

Flow Charts/Diagrams

These scripts run on the same data they ran on before the refactor. No changes there. But for clarity's

sake, I'll reproduce the diagrams/flow charts below:



Documentation

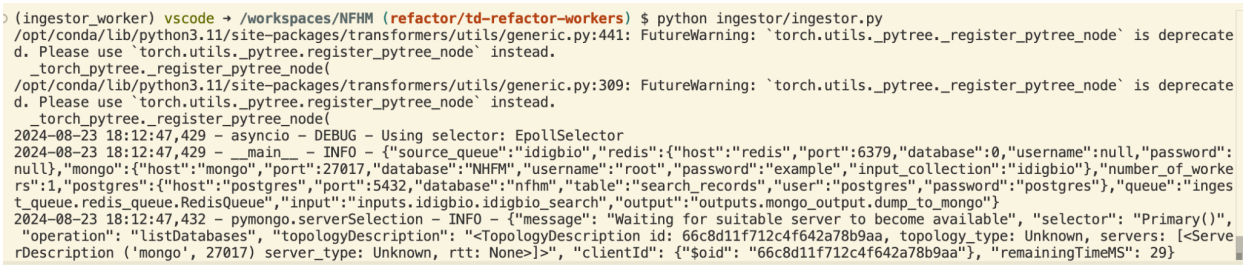
Fortunately, this refactor doesn't change how to run anything. One thing that should be mentioned is that developers should add the Black formatter to their project directory (https://black.readthedocs.io/en/stable/getting_started.html). If they're using VSCode, you can just install the extension, which is probably the easiest way, especially if one configures black formatter to execute on save (https://marketplace.visualstudio.com/items?itemName=ms-python.black-formatter).

Results Visualization

No results to visualize this week.

Proof of Work

The refactor successfully did not alter the behavior of the ingestor scripts. Screencaps show as much:



Next Week's Proposal

- Meeting with Dr Porto and other collaborators at the University of Florida
- Both Roman and Vy snagged on getting Postgres to run their first time setting up the codebase. I'm going to try and fix this next week. Seems to be the only snafu they hit.
- Discuss and possibly start implementing an API accessible version of the ingestor jobs. (E.g., `POST /jobs/vector_embedder { model: "clip_ViT-B-32" }`) to make running jobs and experiments easier.