Week 3 Report

Thomas Deatherage
NFHM/BioCosmos
6 September 2024

Time Slot

1) What progress did you make in the last week?
    - Met with the GaTech Team (Vy, Roman and myself)
    - Met again with UoF collaborators (Dr Porto, Moritz and Chris),
    - Explored techniques for experimenting with different CV/multi-modal models
    - Helped Vy debug her local dev environment
2) What are you planning on working on next?
    - Continue exploring different CV/multi-modal models (bioclip, florence-2, etc.)
    - Depending on when the ingestor team's available (they're in Japan, as I understand), meet with them.
3) Is anything blocking you from getting work done?
    - Looks like the Yale Peabody Museum's servers may be throttling our requests. Not a "blocker", but an inconvenience nonetheless.

Abstracts & Summaries

1) VLM4Bio: A Benchmark Dataset to Evaluate Pretrained Vision-Language Models for Trait Discovery from Biological Images

Conference/Journal: Not stated
Abstract: Images of the natural world collected by a variety of cameras from drones to individual phones are increasingly abundant sources of biological information. There is an explosion of computational methods and tools particularly computer vision for extracting biologically relevant information from images for science and conservation. Yet most of these are bespoke approaches designed for a specific task and are not easily adaptable or extendable to new questions contexts and datasets. A vision model for general organismal biology questions on images is of timely need. To approach this we curate and release TreeOfLife-10M the largest and most diverse ML-ready dataset of biology images. We then develop BioCLIP a foundation model for the tree of life leveraging the unique properties of biology captured by TreeOfLife-10M namely the

abundance and variety of images of plants animals and fungi together with the availability of rich structured biological knowledge. We rigorously benchmark our approach on diverse fine-grained biology classification tasks and find that BioCLIP consistently and substantially outperforms existing baselines (by 17% to 20% absolute). Intrinsic evaluation reveals that BioCLIP has learned a hierarchical representation conforming to the tree of life shedding light on its strong generalizability. All data code and models will be publicly released upon acceptance.
Summary: This paper discusses the importance of using images of nature to gather biological information and the development of a new model called BIOCLIP that can classify different organisms in images more accurately than existing methods.
Link:https://openaccess.thecvf.com/content/CVPR2024/html/Stevens_BioCLIP_A_Vision_Foundation_Model_for_the_Tree_of_Life_CVPR_2024_paper.html
Code: https://imageomics.github.io/bioclip/

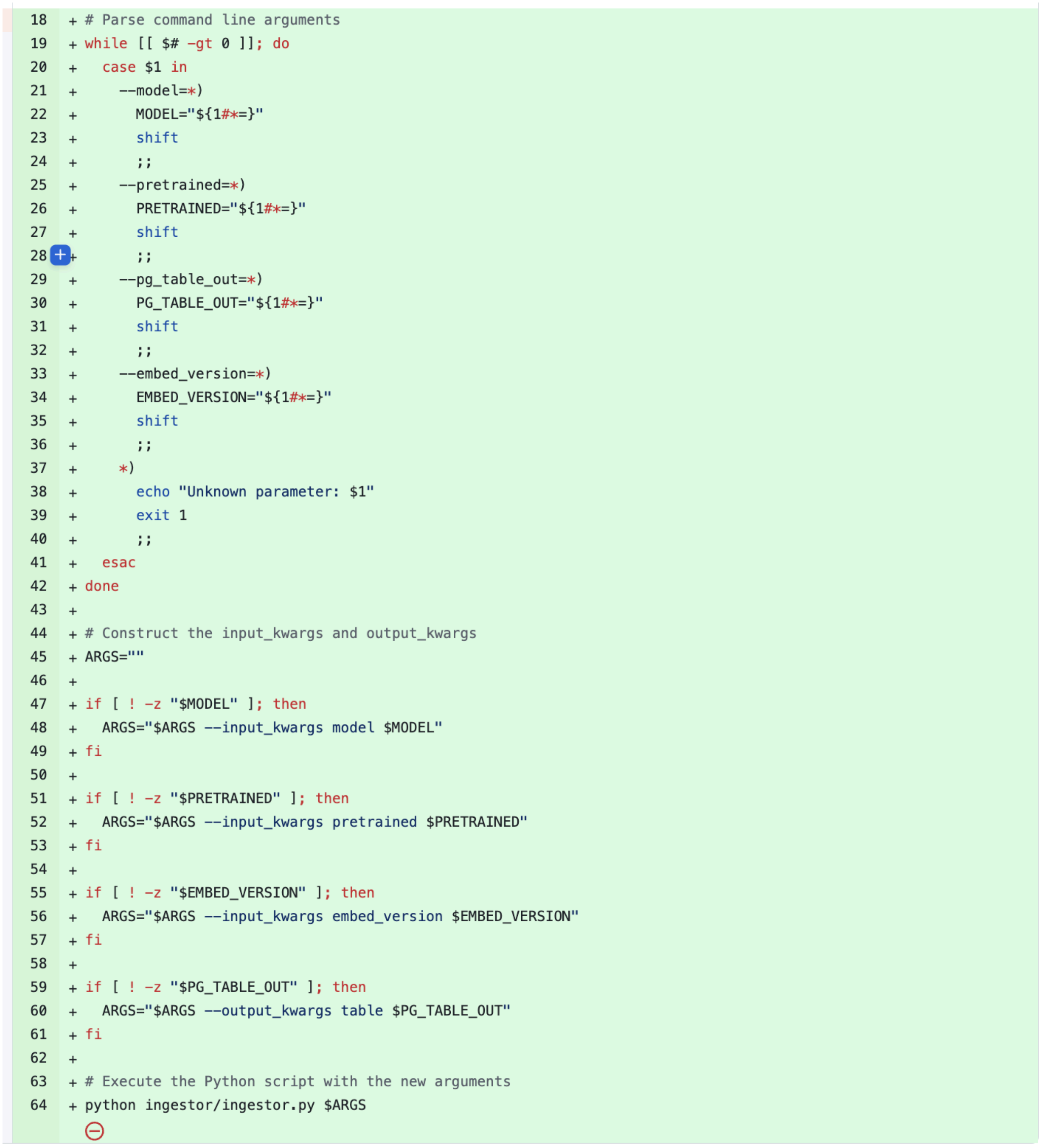
Scripts and Code Blocks

This week I merged my code contributions from last week into the new BioCosmos. See here and here

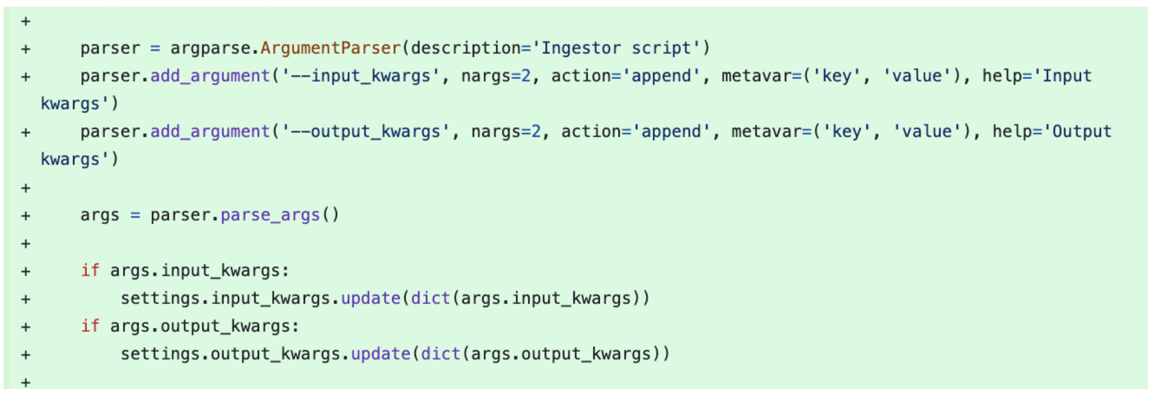
Additionally, I've opened a new PR to the new BioCosmos repo. Per my time slot report above, this week I explored techniques to facilitate experimenting with models. That is to say, I want it to be easy to pick a model, generate embeddings, and casually compare vector search results. Later, I also want to make it easy to formally compare and automatically compare vector search results, perhaps via a tool/dataset like VLM4Bio. Still very much a WIP, my work toward this goal can be found here: https://github.com/BioCosmos-AI/BioCosmos/pull/2/files

Specifically, what I've done so far:
1) Update the embed_ingest job to accept command line arguments for using different OpenClip models and different trained weights
2) Flyway (WIP) integration with postgres to allow for easy versioning/mutation of the Postgres schema/DDL.
3) New columns in the search_records table which will eventually allow API users to specify which embed/experiment version to query their vector search from
4) Minor code improvements, like exponential backoff retries on requests to fetch images from YPM. This is because YPM appears to be throttling our requests to their servers.

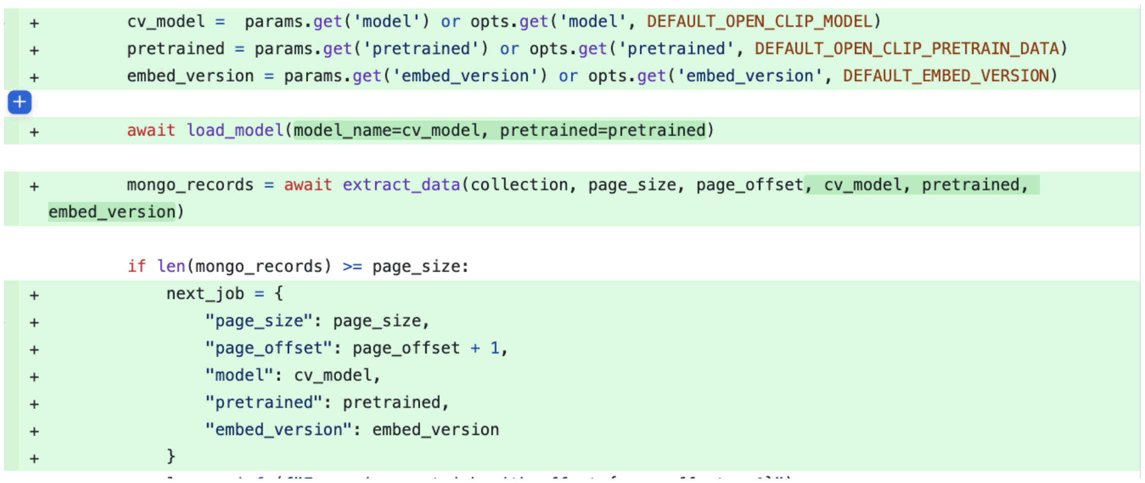
The following three code screenshots show the new ability to specify OpenClip models and pretrained weights from the CLI:



```
+ # Parse command line arguments
+ while [[ $# -gt 0 ]]; do
+   case $1 in
+     --model=*)
+       MODEL="${1#*=}"
+       shift
+       ;;
+     --pretrained=*)
+       PRETRAINED="${1#*=}"
+       shift
+       ;;
+     --pg_table_out=*)
+       PG_TABLE_OUT="${1#*=}"
+       shift
+       ;;
+     --embed_version=*)
+       EMBED_VERSION="${1#*=}"
+       shift
+       ;;
+     *)
+       echo "Unknown parameter: $1"
+       exit 1
+       ;;
+   esac
+ done
+
+ # Construct the input_kwargs and output_kwargs
+ ARGS=""
+
+ if [ ! -z "$MODEL" ]; then
+   ARGS="$ARGS --input_kwargs model $MODEL"
+ fi
+
+ if [ ! -z "$PRETRAINED" ]; then
+   ARGS="$ARGS --input_kwargs pretrained $PRETRAINED"
+ fi
+
+ if [ ! -z "$EMBED_VERSION" ]; then
+   ARGS="$ARGS --input_kwargs embed_version $EMBED_VERSION"
+ fi
+
+ if [ ! -z "$PG_TABLE_OUT" ]; then
+   ARGS="$ARGS --output_kwargs table $PG_TABLE_OUT"
+ fi
+
+ # Execute the Python script with the new arguments
+ python ingestor/ingestor.py $ARGS
```

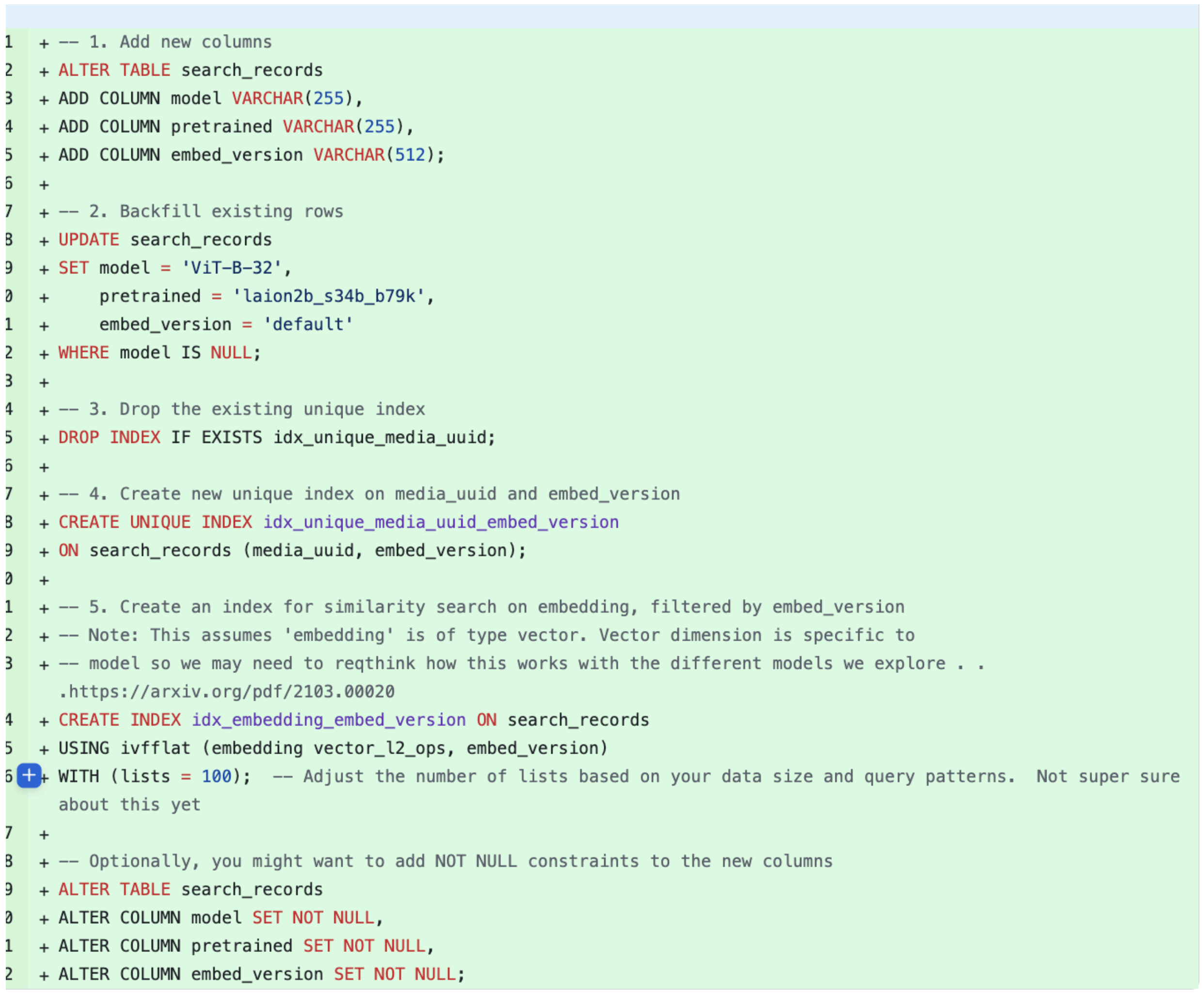


```
+
+     parser = argparse.ArgumentParser(description='Ingestor script')
+     parser.add_argument('--input_kwargs', nargs=2, action='append', metavar=('key', 'value'), help='Input
  kwargs')
+     parser.add_argument('--output_kwargs', nargs=2, action='append', metavar=('key', 'value'), help='Output
  kwargs')
+
+     args = parser.parse_args()
+
+     if args.input_kwargs:
+         settings.input_kwargs.update(dict(args.input_kwargs))
+     if args.output_kwargs:
+         settings.output_kwargs.update(dict(args.output_kwargs))
+
```



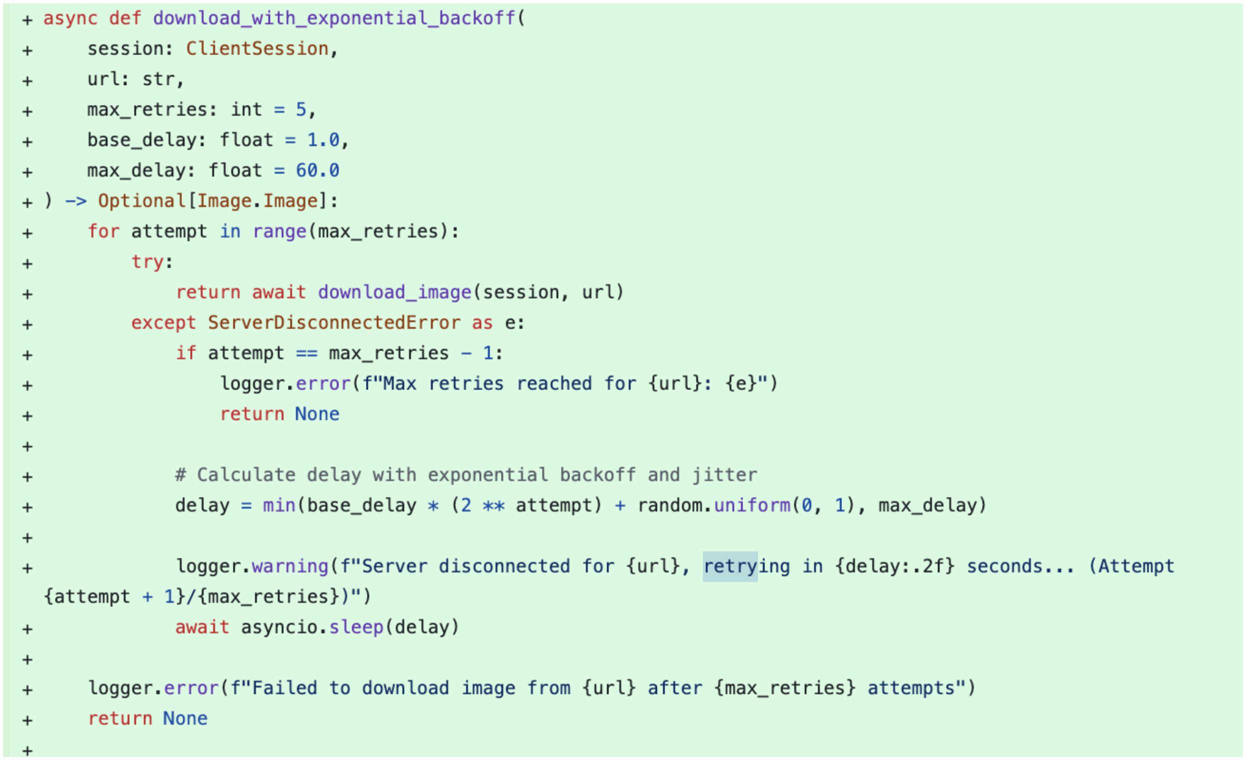
```
+         cv_model =  params.get('model') or opts.get('model', DEFAULT_OPEN_CLIP_MODEL)
+         pretrained = params.get('pretrained') or opts.get('pretrained', DEFAULT_OPEN_CLIP_PRETRAIN_DATA)
+         embed_version = params.get('embed_version') or opts.get('embed_version', DEFAULT_EMBED_VERSION)

+         await load_model(model_name=cv_model, pretrained=pretrained)

+         mongo_records = await extract_data(collection, page_size, page_offset, cv_model, pretrained,
  embed_version)

          if len(mongo_records) >= page_size:
+             next_job = {
+                 "page_size": page_size,
+                 "page_offset": page_offset + 1,
+                 "model": cv_model,
+                 "pretrained": pretrained,
+                 "embed_version": embed_version
+             }
```

Schema changes to Postgres to support (eventually) specifying experiment versions when querying the backend API:



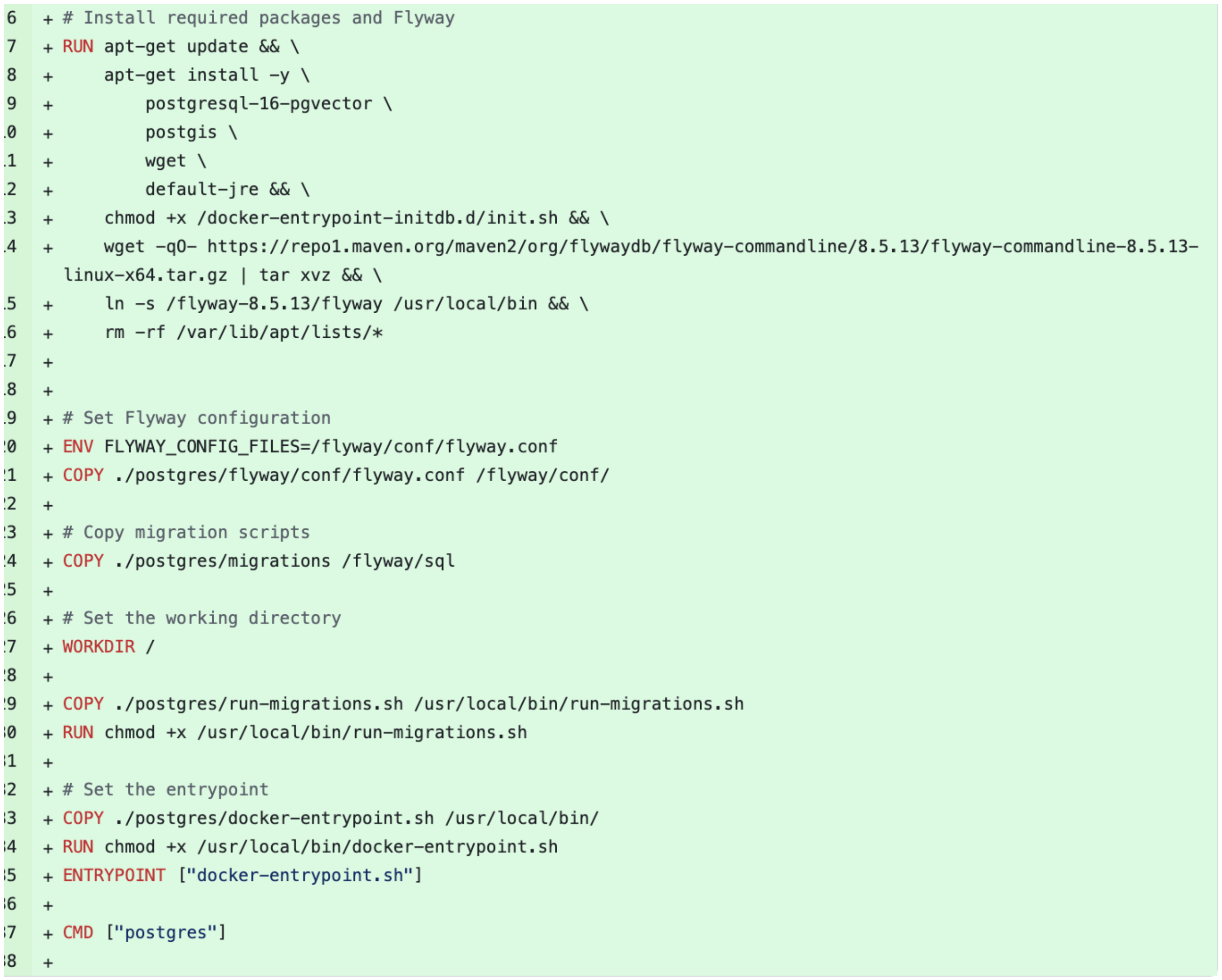
```
+ -- 1. Add new columns
+ ALTER TABLE search_records
+ ADD COLUMN model VARCHAR(255),
+ ADD COLUMN pretrained VARCHAR(255),
+ ADD COLUMN embed_version VARCHAR(512);
+
+ -- 2. Backfill existing rows
+ UPDATE search_records
+ SET model = 'ViT-B-32',
+     pretrained = 'laion2b_s34b_b79k',
+     embed_version = 'default'
+ WHERE model IS NULL;
+
+ -- 3. Drop the existing unique index
+ DROP INDEX IF EXISTS idx_unique_media_uuid;
+
+ -- 4. Create new unique index on media_uuid and embed_version
+ CREATE UNIQUE INDEX idx_unique_media_uuid_embed_version
+ ON search_records (media_uuid, embed_version);
+
+ -- 5. Create an index for similarity search on embedding, filtered by embed_version
+ -- Note: This assumes 'embedding' is of type vector. Vector dimension is specific to
+ -- model so we may need to reqthink how this works with the different models we explore . .
  .https://arxiv.org/pdf/2103.00020
+ CREATE INDEX idx_embedding_embed_version ON search_records
+ USING ivfflat (embedding vector_l2_ops, embed_version)
+ WITH (lists = 100);  -- Adjust the number of lists based on your data size and query patterns.  Not super sure
  about this yet
+
+ -- Optionally, you might want to add NOT NULL constraints to the new columns
+ ALTER TABLE search_records
+ ALTER COLUMN model SET NOT NULL,
+ ALTER COLUMN pretrained SET NOT NULL,
+ ALTER COLUMN embed_version SET NOT NULL;
```

Retry logic with exponential backoff for hopefully handling throttles from YPM. I still need to examine the headers and see if there's any 429 details on when the rate limit will be lifted. If provided, we can be more intelligent in our retry logic.



```
+ async def download_with_exponential_backoff(
+     session: ClientSession,
+     url: str,
+     max_retries: int = 5,
+     base_delay: float = 1.0,
+     max_delay: float = 60.0
+ ) -> Optional[Image.Image]:
+     for attempt in range(max_retries):
+         try:
+             return await download_image(session, url)
+         except ServerDisconnectedError as e:
+             if attempt == max_retries - 1:
+                 logger.error(f"Max retries reached for {url}: {e}")
+                 return None
+
+             # Calculate delay with exponential backoff and jitter
+             delay = min(base_delay * (2 ** attempt) + random.uniform(0, 1), max_delay)
+
+             logger.warning(f"Server disconnected for {url}, retrying in {delay:.2f} seconds... (Attempt
  {attempt + 1}/{max_retries})")
+             await asyncio.sleep(delay)
+
+     logger.error(f"Failed to download image from {url} after {max_retries} attempts")
+     return None
+
```

And part of my yet-unfinished attempt at integrating Flyway into our Postgres Docker container for schema versioning:



```
+ # Install required packages and Flyway
+ RUN apt-get update && \
+     apt-get install -y \
+         postgresql-16-pgvector \
+         postgis \
+         wget \
+         default-jre && \
+     chmod +x /docker-entrypoint-initdb.d/init.sh && \
+     wget -qO- https://repo1.maven.org/maven2/org/flywaydb/flyway-commandline/8.5.13/flyway-commandline-8.5.13-
  linux-x64.tar.gz | tar xvz && \
+     ln -s /flyway-8.5.13/flyway /usr/local/bin && \
+     rm -rf /var/lib/apt/lists/*
+
+
+ # Set Flyway configuration
+ ENV FLYWAY_CONFIG_FILES=/flyway/conf/flyway.conf
+ COPY ./postgres/flyway/conf/flyway.conf /flyway/conf/
+
+ # Copy migration scripts
+ COPY ./postgres/migrations /flyway/sql
+
+ # Set the working directory
+ WORKDIR /
+
+ COPY ./postgres/run-migrations.sh /usr/local/bin/run-migrations.sh
+ RUN chmod +x /usr/local/bin/run-migrations.sh
+
+ # Set the entrypoint
+ COPY ./postgres/docker-entrypoint.sh /usr/local/bin/
+ RUN chmod +x /usr/local/bin/docker-entrypoint.sh
+ ENTRYPOINT ["docker-entrypoint.sh"]
+
+ CMD ["postgres"]
+
```

Flow Charts/Diagrams

Nothing new flow-chart wise to show just yet.

Documentation

The primary documentation changes for this week are that one can pass model and pretrained weights parameters to the ingest_embed job. For example:

$ bin/ingest_embed --model="Vit-B-32" --pretrained="laion2b_s34b_b79k"

Results Visualization

No results to visualize this week.

Proof of Work

Still a WIP as of this week. Should have the end-to-end flow working by next week and can provide a proof of it then.

Next Week's Proposal

- Finish the model-selection support I started this week.
- Meet with Roman and Vy (and optionally Bree and Dr Porto) as usual on Wednesday.
- Meet with UoF collaborators, depending on schedules