Week 4 Report

Thomas Deatherage
NFHM/BioCosmos
13 September 2024

Time Slot

1) What progress did you make in the last week?
    - Met with the GaTech Team (Vy, Roman and myself on Wednesday evening)
    - Met with GaTech + UoF collaborators.
    - Continued last week's WIP (DB versioning + embedding tags for segmenting experiments)
2) What are you planning on working on next?
    - Per today's meeting, I'm going to focus on Florence-2's trait grounding and trait referring capabilities.
3) Is anything blocking you from getting work done?
    - I'll be traveling for a wedding the latter half of the week, so I may not have a report for next week.

Abstracts & Summaries

1) Re-TASK: Revisiting LLM Tasks from Capability, Skill, and Knowledge Perspectives

Conference/Journal: Not stated
Abstract: As large language models (LLMs) continue to scale, their enhanced performance often proves insufficient for solving domain-specific tasks. Systematically analyzing their failures and effectively enhancing their performance remain significant challenges. This paper introduces the Re-TASK framework, a novel theoretical model that Revisits LLM Tasks from cApability, Skill, Knowledge perspectives, guided by the principles of Bloom's Taxonomy and Knowledge Space Theory. The Re-TASK framework provides a systematic methodology to deepen our understanding, evaluation, and enhancement of LLMs for domain-specific tasks. It explores the interplay among an LLM's capabilities, the knowledge it processes, and the skills it applies, elucidating how these elements are interconnected and impact task performance. Our application of the Re-TASK framework reveals that many failures in domain-specific tasks can be attributed to insufficient knowledge or inadequate skill adaptation. With this insight, we propose structured strategies for enhancing LLMs through targeted knowledge injection and skill adaptation.

Specifically, we identify key capability items associated with tasks and employ a deliberately designed prompting strategy to enhance task performance, thereby reducing the need for extensive fine-tuning. Alternatively, we fine-tune the LLM using capability-specific instructions, further validating the efficacy of our framework. Experimental results confirm the framework's effectiveness, demonstrating substantial improvements in both the performance and applicability of LLMs.

Summary: This paper presents the Re-TASK framework, which helps improve large language models (LLMs) for specific tasks by analyzing their capabilities, knowledge, and skills. It identifies reasons for their failures and suggests targeted strategies to enhance their performance without needing extensive adjustments.

Link: https://arxiv.org/abs/2408.06904

Code: N/A

Relevance: Part of our long term vision is to have an LLM interface capable of delegating queries to appropriate, well-defined tasks. This paper is part of my research into that.

Scripts and Code Blocks

Code contributions can be found at this PR: https://github.com/BioCosmos-AI/BioCosmos/pull/2

This week is a continuation of last week. Primary accomplishments:

1) Finally got DB versioning working with Flyway. I'm going to have Roman or Vy test this on their machines before I merge.
2) Support end-to-end arbitrary tagging from the generate-embedding stage to API search

Some highlights:

```
+ #!/bin/bash
+ set -e
+
+ # Function to check if PostgreSQL is ready
+ pg_isready() {
+     PGPASSWORD=$POSTGRES_PASSWORD psql -h localhost -U "$POSTGRES_USER" -d "$POSTGRES_DB" -c "SELECT 1" > /dev/null 2>&1
+ }
+
+ # Start PostgreSQL
+ /usr/local/bin/docker-entrypoint.sh postgres &
+
+ # Wait for PostgreSQL to be ready
+ until pg_isready; do
+   echo "Waiting for PostgreSQL to be ready..."
+   sleep 2
+ done
+
+ echo "PostgreSQL is ready. Creating extensions..."
+
+ # Create extensions
+ psql -v ON_ERROR_STOP=1 --username "$POSTGRES_USER" --dbname "$POSTGRES_DB" <<-EOSQL
+     CREATE EXTENSION IF NOT EXISTS postgis;
+     CREATE EXTENSION IF NOT EXISTS vector;
+ EOSQL
+
+ echo "Extensions created. Running Flyway migrations..."
+
+ # Run Flyway migrations with baseline
+ flyway -url=jdbc:postgresql://localhost:5432/$POSTGRES_DB -user=$POSTGRES_USER -password=$POSTGRES_PASSWORD -baselineOnMigrate=true -locations=filesystem:/flyway/sql migrate
+
+ echo "Flyway migrations completed."
+
+ # Keep the container running
+ wait
```

New entrypoint into Postgres

```python
@router.post("/search", response_model=SearchResponse)
async def search(
    request: Request,
    search_param: Optional[str] = Form(None),
    image: Optional[UploadFile] = File(None),
    limit: int = Query(30, ge=1, le=100),
    embed_version: str = Query("default", max_length=512),
    session: AsyncSession = Depends(get_session)
) -> SearchResponse:
    try:
        if search_param is None and image is None:
            raise HTTPException(status_code=400, detail="At least one of search_param or image must be provided.")

        search_vector = await process_input(request, search_param, image)

        query = select(SearchRecord).where(
            SearchRecord.embed_version == embed_version
        ).order_by(
            func.l2_distance(
                SearchRecord.embedding,
                cast(search_vector, Vector)
            )
        ).limit(limit)

        results = await session.execute(query)
        records = results.scalars().all()

        payload = [create_record_payload(record) for record in records if record.external_media_uri is not None]

        return SearchResponse(
            search_param=search_param,
            filename=image.filename if image else None,
            record_count=len(payload),
            records=payload,
            embed_version=embed_version
        )
```

The /search endpoint now accepts a query parameter “embed_version” to filter tagged rows.

Flow Charts/Diagrams

Nothing new flow-chart wise to show.

Documentation

I added documentation to the repo’s README

Database Versioning.

We use Flyway to version the Postgres DB. To make a new migration/version, add your migration changes to a file named per the Flyway Naming Conventions to the `postgres/migrations` directory. Then rebuild the docker container.

Running and tagging experiments

When generating embeddings, you have the option to tag rows in Postgres with an arbitrary string of text. This is useful for running experiments and comparing results. The string will be stored in the Postgres database and can be used to identify the embedding later.

Example

```
$ bin/ingest_embed --embed_version "thomas_experiment_22" --pretrained "ViT-B-32" --model "laion2b_s34b_b79k"
```

- From the workbench of Redis Insight, pass a simple search string to the `embedder` queue:
  - `LPUSH embedder "{}"`

From the API, you can limit results to a specific "embed_version" tag:

```
curl -X 'POST' \
  'http://localhost:8080/api/search?limit=30&embed_version=thomas_experiment_22' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'search_param=butterfly'

{
  "search_param": "butterfly",
  "filename": null,
  "record_count": 100,
  "records": [. . .],
  "embed_version": "thomas_experiment_22"
}
```

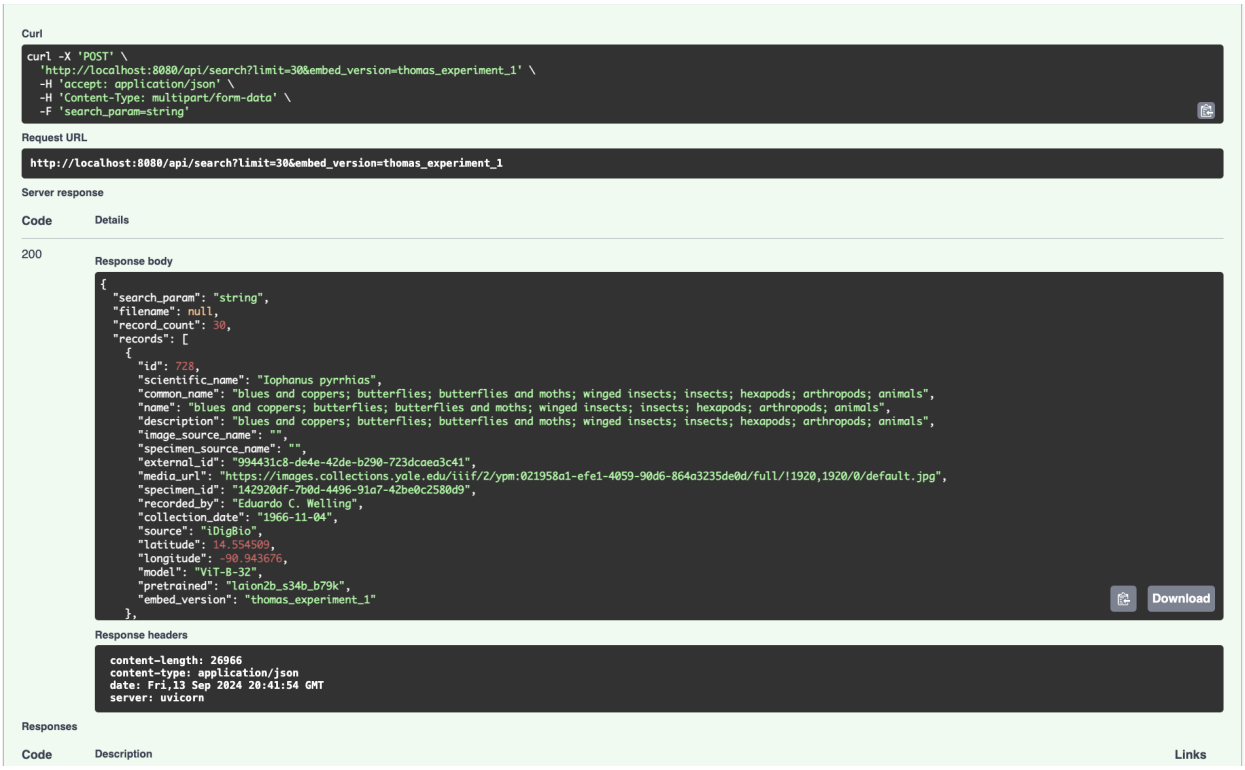
Note this only works for Open CLIP gennerated embeddings at the moment. (TODO: Update search query to select correct embedding model by "embed_version").

Results Visualization + Proof of Work

When generating embeddings, we can now tag an arbitrary “embed_model”.

```
$ bin/ingest_embed --embed_model=thomas_experiment_1
```

And on the read-side of the equation, we can filter our focus to a specific tagged version of generated embeddings:



Next Week's Proposal

- Merge this week's work
- Focus on Florence-2 and training groundings/referrings