CriticalMAAS Milestone 4 report – Macrostrat TA4 team

Daven Quinn and the UW–Madison / Macrostrat CriticalMAAS team

April 1, 2024

Contents

1 Report period

This **Milestone 4** report describes the the technical progress of the UW–Madison – Macrostrat team on the CriticalMAAS project, from February 23–March 29 (Month 7). It builds on the research and technical objectives reported in our Phase 1 Research Plan and developments since submitting our Milestone 3 report in late February 2024.

2 Code and documentation

The primary deliverable for CriticalMAAS Milestone 4 is a second release of code and documentation covering Macrostrat's contribution to CriticalMAAS. These components are indexed in the **UW–Macrostrat CriticalMAAS README** at the root of the the UW-Macrostrat/CriticalMAAS GitHub repository.

Macrostrat is a multi-faceted system, and building new functionality for CriticalMAAS involves many codebases, which are summarized below. Documentation is available at Macrostrat documentation website (*in early development*) and in each repository's `README.md` file.

2.1 Macrostrat core system

A modular Macrostrat system, the key code deliverable of Macrostrat's CriticalMAAS work, is housed in the UW-Macrostrat/macrostrat repository. This codebase includes Macrostrat's database schema, management scripts, control framework, and ingestion pipeline. Several other repositories contain important elements of the Macrostrat system, including:

- UW-Macrostrat/macrostrat-api: Macrostrat's current (v1-2) API, which provides access to Macrostrat's web services for current functionality
- UW-Macrostrat/tileserver: Server for vector and raster tiles to GIS software and TA3 performers
- UW-Macrostrat/api-v3: Macrostrat's v3 API, which will be the primary API for new capabilities, including CriticalMAAS
- UW-Macrostrat/python-libraries: Shared Python libraries for database access and application management.
- UW-Macrostrat/web: Macrostrat's main web interface codebase
- UW-Macrostrat/web-components: Shared React/Typescript components for geological user interfaces

2.2 Geologic metadata curation

Several codebases are being developed to curate rock-record descriptions from the geological literature through relationship extraction and named entity resolution atop the xDD platform. Some of these predate CriticalMAAS but are being evaluated to drive program functionality.

- UW-Macrostrat/unsupervised-kg: Unsupervised knowledge graph construction to discover new entities and relationships from geological literature.
- UW-Macrostrat/llm-kg-generator: LLM-assisted graph generation to extracts geological facts, operating over the scientific literature to characterize batches of geologic names.
- UW-xDD/text2graph_llm: A tool to transform textual data into structured graph representations, using LLMs to identify and extract relationships between locations and geological entities from text.

- UW-Madison-DSI/ask-xDD: A chat interface and API endpoint for accessing academic information via Retrieval-Augmented Generation (RAG). The prototype currently covers topics such as geoscience, climate change, and COVID-19.

Work is just beginning to integrate these tools into an infrastructure for bulk extraction of geologic information, which will be housed at UW-Macrostrat/macrostrat-xdd.

2.3 Geologic map editing

A prototype human-in-the-loop (HITL) geologic map editing system is being developed in several repositories:

- Mapboard/topology-manager: a topological map manager for geologic maps stored in PostGIS.
- Mapboard/Mapboard-Platform: a web infrastructure to support geologic map editing.

Both of these are Apache 2.0 licensed and rely on Macrostrat's Python libraries. Maps in the system will be editable through standard GIS platforms; the natural drawing interface of the **Mapboard GIS** app will also be available but optional, as its codebase is not open-source (see Sec. 3.4).

2.4 Program coordination

Macrostrat has contributed code for shared infrastructure for the CriticalMAAS program, including data formats, schemas, and shared libraries.

- DARPA-CriticalMAAS/ta1-geopackage: A GeoPackage-based data format for validating and storing TA1 output. It is unclear if this will be used in the final system, but it was created to support the program before the CDR took on the responsibility of data storage and schema conformance.
- DARPA-CriticalMAAS/schemas: A repository for schemas and data formats for TA1-3 integrations (*started by UW–Macrostrat and subsequently contributed to by all TA teams*)
- UW-xDD/document-store: A supplemental store for public/user provided PDFs that provides full-text access, integrates with xDD APIs. **Note: This repository is being integrated into the CDR codebase.**

3 Technical accomplishments

3.1 Macrostrat core system

Significant progress has been made on consolidating Macrostrat's core system into a more declarative and unified framework. Key accomplishments include

- **All Macrostrat core codebases are now publicly available under the *Apache 2.0* license.** This is a major milestone in our fulfillment of the CriticalMAAS requirements for a system deliverable to USGS.
- Macrostrat's control framework has been considerably improved in this code release, with a `macrostrat` command line interface with subcommands for database management and map ingestion.
- Automated test suites and CI pipelines have been added to the Macrostrat Python libraries and v3 API to ensure stable capabilities.
- Most Macrostrat services have been containerized for local or cloud deployment, with CD processes that track both the `main` branch and tagged releases.

Work is ongoing to fully document the Macrostrat codebase, system design, capabilities, and usage. A new documentation website available at dev2.macrostrat.org/docs to centralize these resources. This site is still in early development, but its early form links to resources such as API documentation (OpenAPI-compatible for new APIs), example user interfaces, and documentation for shared Python and web component libraries. The CriticalMAAS-specific entry-point into the documentation, the **UW–Macrostrat CriticalMAAS README**, has been updated with links into specific parts of this documentation footprint.

Most of the technical pieces are in place to run a fully functional standalone deployment from `macrostrat up`. Several major missing elements are the focus of current development:

- The system requires an existing Macrostrat database dump with at least some data pre-filled (especially data dictionaries). One key problem that some data models refer to vocabulary elements by their integer primary key, which makes portability between Macrostrat instances challenging. This issue is already being encountered between Macrostrat's development and production databases, underscoring the need for a solution.
- Macrostrat's legacy API (v1-2) still maintains a dependency on MariaDB for some data routes. The PostGIS database that forms the core of Macrostrat's streamlined implementation contains all data, but work is needed to port queries and ensure that the new API does not break existing functionality. We are currently reworking the API's testing framework (which has been inoperative for several years) and will use that to ensure conformance and compatibility.

Our goal is for a standalone deployment of Macrostrat including these elements to be possible by soon after the 9-Month Hackathon, to establish a well-documented and tested process by the end of the program.

Map ingestion pipeline

Progress on the since Milestone 3 has been primarily focused on consolidating the map ingestion pipeline to be more adaptable and flexible, in response to workflow deficiencies identified at the 6-month Hackathon. We've significantly refined the command-line interface to allow maps to be staged, prepared, and finalized using a unified interface. Several issues limiting the functionality of legend curation tools have been resolved, and the legend preparation web interface has been implemented for points and lines (initial development focused on map

polygons). Additionally, a new metadata management interface is being developed to allow the progress of maps through the ingestion pipeline to be tracked and managed. This interface will streamline the curation of large numbers of maps and allow for the easy identification of maps that require additional attention Sec. 4.2. A key target is to have this tool ready for USGS evaluation and testing by the 9-month Hackathon.

Provider-side filtering of geologic maps for TA3

One of the major gaps identified at the 6-month Hackathon was the lack of server-side capabilities to filter Macrostrat's vector tiles. This gap was a result of TA3's expected usage pattern changing substantially to a server-side-filtering approach, which was not anticipated as being a key need based on *Lawley et al., 2023* modeling approaches. We have made substantial progress on scoping and designing this feature, and will implement it prior to the 9-month Hackathon. This new feature will be developed alongside the standardization of a CI/CD pipeline for the Macrostrat tileserver codebase, bringing it into line with our other services.

Filter query language Open-ended filtering is difficult functionality to implement in Macrostrat's API due to the number and heterogeneity of data fields and the prospect of complex queries across different regions and maps. We are hoping to implement this functionality using the semantics of an existing query language to improve usability and clarity. We are currently evaluating the potential of several query languages to drive API filtering. These include

- Direct specification of a WHERE clause as a query parameter. This approach is similar to that used by the ArcGIS Online platform but will be difficult to document. The encoding of spatial operators may also be challenging.
- PostgREST-style filtering, which faithfully exposes the filtering capabilities of PostgreSQL in a small number of query parameters. However, it does not support spatial operators, which are difficult to forgo for the CriticalMAAS use case.
- "Common Query Language" (CQL) filtering, which is a standard for querying geospatial data in development by the Open Geospatial Consortium (OGC). We are evaluating the pygeofilter library to help implement this functionality.

Overall, the balance of usability, expressiveness, and documentation suggest that the CQL approach is the most promising. This will be the focus of our prototyping going forward.

Querying Macrostrat's lithology hierarchy One other usability gap we are working to address is how Macrostrat's lithology dictionaries are exposed for API querying. Macrostrat hosts tree-based lithology taxonomy that is similar (and conformant to) existing dictionaries such as the USGS lithologic classification and the State Geologic Map Compilation lithologic hierarchy. We track rock types in a four-level, denormalized tree, `lith -> group -> type -> class`. For instance,
https://dev2.macrostrat.org/api/v2/defs/lithologies?lith=basalt

yields
`basalt -> mafic -> volcanic -> igneous`. This taxonomy can be accessed in the Macrostrat API and web application.

Unfortunately, querying this tree in Macrostrat's current API is somewhat awkward, requiring separate queries at each level of the hierarchy. For instance, you must know a priori that `mafic` is a `lith_group` and query as such, `../lithologies?lith_group=mafic`. `../lithologies?lith=mafic` would return a different, more restricted set of results, missing terms encompassed in the `mafic` group like `basalt`. This makes it difficult to construct queries naturally, as you have to know the specific level of each term. As part of designing tileserver filtering, we are hoping to improve this situation by allowing queries expressed in terms of a single lithology to be automatically expanded to include all child terms.

- A query for `lith=volcanic` would automatically include `rhyolite`, `basalt`, and other extrusive igneous rocks.
- The ability to return exact matches will be preserved with a `lith:exact` parameter.

This will make it easier to construct complex lithological queries and preserve the option to add new levels to the lithology hierarchy in the future.

3.2 Document store and CDR integration

The CriticalMAAS Document Store was initially produced to host program-relevant documents found in xDD for more efficient access by the CriticalMAAS program, without an ongoing dependency on xDD. This capability is identical (by design) to the needs of the TA2-supporting CDR, and we are working with Jataware to merge the document store into the CDR codebase. This month, in addition to working to plan this integration, we have also added models and basic API routes to hold page extractions (e.g., identified bounding-boxes of figures, tables, and other document elements). Once this integration is complete, the CDR will assimilate the ~80,000 documents currently hosted by the document store, along with COSMOS entity extractions for many document components. The document store will then be retired as a standalone codebase.

3.3 Geologic metadata curation

We are working towards building better rock-record descriptions from the geological literature by:

- Discovering concepts linked to known geological units
- Finding new units based on proximity to known entities

These capabilities will allow structured data describing rock units to be extracted from xDD and used to populate Macrostrat's lexicon. This will allow straightforward searching of unit descriptions extracted from the geologic literature, using xDD in a batch, offline mode.

We are beginning to test these pipelines in bulk, build a more robust infrastructure to support them, and build towards both batch extraction of geologic information from xDD and HITL

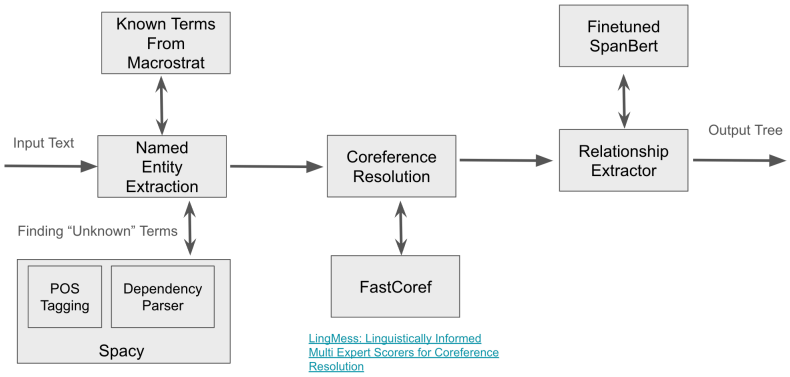

Figure 1: System diagram of the unsupervised knowledge graph construction pipeline

feedback tools to curate the resulting dataset.

Unsupervised knowledge graph construction

The `UW-Macrostrat/unsupervised-kg` system (*Devesh Sarda*; Computer Science) uses unsupervised knowledge graph construction to discover new entities from the geological literature. This system includes separate components for named entity resolution, coreference resolution, and relationship extraction. It produces highly curated graphs, but these often lack sensitivity to nuanced natural language patterns governing descriptions. This system has been evaluated end-to-end over several thousand documents, with the results available in Macrostrat's prototype web viewer.

LLM-assisted relationship extraction

The other approach, housed in `UW-Macrostrat/llm-kg-generator`, extracts geological facts using LLM prompts to produce graph of related terms, and filters these graphs by semantic similarity to known entities.

This system has been tested over 5000 documents, with 4,151 stratigraphic names and 7,123 different descriptive terms extracted. Current work is focused on filtering these attributes against Macrostrat's ~47,000 known stratigraphic names, 212 lithologies and several hundred lithology attributes. Semantically meaningful links to other terms will be preserved as well. A multi-stage prompting approach is used.

1. Identify and extract entities from the text using exact matching against a Macrostrat known entities.
2. Include the details of these identified entities in the input prompt, instructing the language model to consider these entities while determining relationships.
3. Extract relationship triplets from the LLM on the enriched prompt.
4. Refine the extracted entities by comparing their semantic similarity to the known entities specified in the original query.

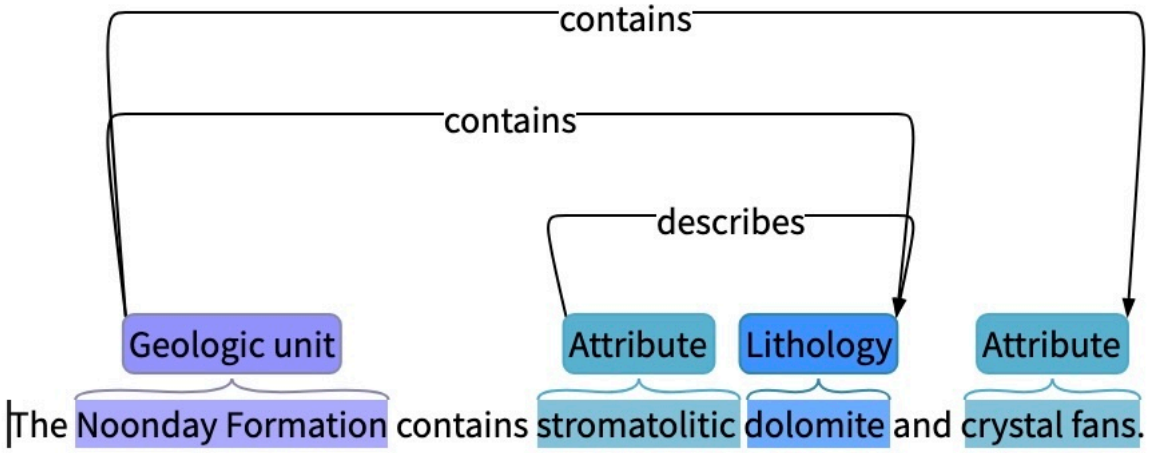

Figure 2: Demo editable display of attributes for the Noonday Formation based on the Poplar NER tagging tool

5. Incorporate standardized Macrostrat terms and IDs for the known entities into the final relationship triplets.

Infrastructure and feedback

Each of these models is designed for unattended characterization of a geologic unit against its particular footprint in the published geologic literature. They will be run in an offline batch mode against xDD against Macrostrat's existing lexicon of geologic names. An infrastructure (`UW-Macrostrat/macrostrat-xdd`; *placeholder*) is being developed to deploy these models, standardize results, link them to data dictionaries, and store them in Macrostrat's PostgreSQL database as "candidate" lithologic descriptions. The goal of this infrastructure will be to create bulk descriptions of geological entities from xDD that can be validated and refined by human curators and included in Macrostrat's data dictionaries and filtering systems.

A HITL feedback interface (currently in development; see early prototype) will allow users to view and correct extracted rock descriptions, as well as establish *de novo* relationships. We have been evaluating off-the-shelf NER tagging tools to identify user-interface components that can form the core this feedback system in Macrostrat's existing web interface. While some commercial and open-source tools (e.g., Prodigy, Label Studio, Doccano, and Poplar) are promising, it is likely that we will customize an existing solution for use within our web interfaces. Initial work is ongoing in the `Macrostrat/web-components` repository. These components will be structured as a standalone library so that it is easy to implement NER tagging for other problems, including potential TA2 feedback uses, that may arise.

3.4 Geologic map editing

We have demonstrated prototype functionality for editing geologic maps at the Month 3 and 6 hackathons. Our approach is based on the `Mapboard/topology-manager` package and the

Mapboard GIS iPad app. This month, we have made substantial progress in bringing this capability into CriticalMAAS data, in order to allow quick feature digitization and correction of TA1 outputs.

The geologic map system is designed to make topological solving iterative, allowing a correct map to be built from a series of small edits. This will allow maps to be corrected by simple operations like moving a line or changing a unit, rather than requiring topologically complex operations like merging polygons.

This month, we streamlined the core `Mapboard/topology-manager` codebase, shifting its orchestration code from Node.js to Python to match the rest of the Macrostrat infrastructure. This codebase now includes a full suite of tests and a CI pipeline, is able to work with multiple maps in the same database, and can now be invoked programmatically from Python instead of just as a watcher daemon. This new flexibility will allow the system to be used in a wider variety of contexts, including prioritizing overlapping maps in Macrostrat's core system (solving Macrostrat's topology is currently a "bulk" process that takes over 9 hours to run on all maps; an iterative approach promises to speed this by orders of magnitude).

The orchestration interface for the map editing system is being developed in the `Mapboard/Mapboard-Platform` repository, using a similar toolkit to the Macrostrat control scripts that will allow easy integration in the future if desired.

Since the map editing system is based on PostGIS, it can be operated on by a wide variety of GIS editing tools, particularly QGIS. However, the system is designed to function best when augmented by natural editing capabilities provided by the **Mapboard GIS** iPad app and server components. These capabilities can be integrated with the CriticalMAAS system to speed editing, including in a web interface, but they **are not available under an open-source license**. We should discuss this with DARPA and USGS, but if these capabilities cannot be used for that reason, direct connection via GIS and/or simplified editing APIs and web interfaces can be prioritized.

4 Issues and Concerns

4.1 "Structurally complete" TA1 maps

Between the 3-Month and 6-Month Hackathons, we created the TA1 Geopackage Format in conversation with TA1 performers. This format had two goals:

1. The pre-CDR integration design prioritized exchange of flat files between teams, rather than the webhook architecture adopted in the CDR. A "self-describing" format ensured that TA1 outputs could be validated at the point of creation.
2. The format was intended to encompass a "structurally complete" geologic map, with all necessary metadata and relationships for downstream use. This was intended to ensure that necessary evaluation, validation, and correction occurred in the TA1 phase through HITL tools, and that the endpoint of TA1 was a complete, validated geologic map packaged in one file.

3. The format was intended to be easily openable in GIS software, allowing quick evaluation of the product without custom tooling.

Since the CDR has taken on the responsibility of data storage and schema conformance, the TA1 Geopackage has been rendered largely unnecessary as a data-passing format. But the prioritization of more "atomic" TA1 extractions in the CDR raises the risk that these outputs will be evaluated piecemeal and never integrated into a complete geologic map. The compositing of TA1 outputs into a full representation of a map is a complex problem in which both TA1 performers (who know the relative strengths and interactions between their models) and USGS staff operating HITL interfaces need to participate. If this integration exercise falls by the wayside, the burden of deciding between different TA1 outputs will be pushed onto the analysts doing a mineral assessment. This will either slow analysts substantially, requiring them to repeatedly evaluate the map extractions for each new mineral modeling task, or require new, complex probabilistic modeling approaches to integrate multiple potential maps into TA3 analytical pipelines. Developing a clear roadmap around the structure and capabilities of HITL tools, workflows, and CDR formats targeting TA1 integration will be necessary to address this issues (Sec. **??**).

4.2 Availability of vector maps

In our Milestone 3 report, we reported difficulty of accessing vector maps for ingestion into Macrostrat. This concern is unchanged. As we build a map ingestion pipeline, access to vector maps is a key bottleneck to integrating modern, "born-digital" mapping into TA3 modeling workflows. We have made some progress in acquiring datasets from state geological surveys, but these products of are of variable quality (e.g., some maps digitized by the Nevada Geologic Survey lack even cursory unit descriptions). High quality datasets have to be actively and systematically sought and acquired.

At the Month 6 hackathon, we discussed the possibility of engaging with Jataware to use their web scraping system to find and stage vector maps (possibly using the CDR as a indexing system). This would be a significant improvement over current practice, which requires manually finding and staging maps or writing our own web-scraping scripts that retread similar ground to Jataware's exercise for raster maps. Jataware's focus is currently on building the CDR, so this has fallen by the wayside.

Even better would be working with the USGS to integrate their vector map collection, which is maintained internally by NGMDB in relatively "clean" data formats (at least in some cases). A systematic survey of available geologic maps would be akin to the exercise being undertaken for geophysics under EarthMRI. However, this has been off the table due to organizational barriers within USGS. Given that vector mapping compiled from the 1990s onwards tends to be the "best" geologic mapping data where available, CriticalMAAS is at a significant disadvantage for its inability to access this resource. Some high-level attention to this problem between DARPA and the USGS divisional leadership would be beneficial to ensuring that the CriticalMAAS program can access the datasets needed to deliver its full suite of capabilities.

4.3 Development roadmap clarity

The program transition to the CDR and a new integration structure has been challenging to navigate as a TA4 performer. We have worked to build tools (e.g., the TA1 Geopackage Format) with usefulness to the program, but such program-level functionality now must be closely coordinated with the CDR. It is unclear what the CriticalMAAS program's expectations are for the development of HMIs and other user-facing tools in this new program design. We need to clearly communicate with DARPA and Jataware about the specific development roadmap in order to not waste effort building unnecessary functionality.

Laudably, Jataware is implementing the CDR with an open design atop which "anyone can build a HMI." This is a good principle to guide CDR design and ensure flexibility (indeed, Macrostrat's APIs generally take the same approach). However, "anyone can build anything" is not a useful approach to planning work between performers. We don't have a lot of insight into the envisioned design: for instance, will be some sort of a "CriticalMAAS Portal" entry-point for searching/interacting with the CDR and its holdings? If so, presumably Jataware will decide how that is oriented and build most of its functionality. We have not received clear guidance on the intention here despite pressing for specifics (see Sec. 4.1).

Overall, Macrostrat and Jataware are the two organizations with expertise to build HMIs for program integration. Macrostrat doesn't have a pool of staff that can surge into a project and accomplish a lot of HMI work rapidly, so we need to approach development responsibilities with a clear roadmap and division of effort. To that end, it's imperative that we do some TA4-level planning and horse trading around "who's building what" early enough in the program that we can actually deliver capabilities atop the CDR if necessary. We will continue to push Jataware to commit to specifics but hope that DARPA can also put a focus on concretizing plans for how users will interact with the CriticalMAAS system as a whole.