

GCN Optical Extraction

LLM-based parser, regex baseline, and MCP server for GCN Circulars — project plan and TODO

This document captures everything required to deliver the optical-extraction phase of the GCN parser project: scope, target schema, comparison methodology, concrete work breakdown, and the decision log from initial advisor scoping. It is organized so each section can be checked off independently.

Deliverable in one sentence. An LLM-based extractor that converts the free text of GCN optical circulars into structured JSON conforming to the official GCN schema, with a serious regex baseline for comparison, exposed as an MCP server that SkyPortal can consume.

Starting point	New repo; attribution to github.com/sjhend03/GCNMCP in README
Output schema	github.com/nasa-gcn/gcn-schema (extend Photometry, add 2 new schemas)
Vocabulary	github.com/skyportal/timedomain-taxonomy (classification controlled list)
Eval + archive	github.com/nasa-gcn/circulars-nlp-paper (Vidushi's labeled set + full archive)
Background	Sharma et al. 2025 — arxiv.org/abs/2511.14858
Eventual consumer	SkyPortal, via MCP server + chatbot frontend

1. Project Goals

The GCN Circulars archive contains over 41,000 human-written observation reports spanning nearly 30 years. The unstructured prose makes systematic information extraction difficult. The goal of this project is to build an LLM-based extractor that converts circular text into structured records, prove it outperforms a regex baseline, and expose it through an MCP server so SkyPortal (and other consumers) can pull structured information in real time.

What's already in hand

The `nasa-gcn/circulars-nlp-paper` repo (Sharma et al. 2025) ships substantial scaffolding for this project: the full circulars archive as JSON through May 2025, topic labels for all 40,507 circulars (including a clean "Optical Observations" subset), a 13,593-row labeled redshift dataset with Vidushi's predictions side-by-side with Swift catalog ground truth, and her information-extraction notebook with the actual prompts she used. This collapses two weeks of hand-labeling on redshift, GRB number, telescope name, and redshift type — and gives a published baseline to beat. The other ~20 fields the advisor listed (full photometry, astrometry, spectroscopy lines, classification, etc.) still need labels.

Why optical first

The advisor's instruction is to start with optical because that is where regular expressions will fail hardest: multi-row magnitude tables, magnitude-system inference (AB vs Vega vs ST), and in-prose classifications. Concentrating on optical maximizes the LLM's measurable advantage in the regex-vs-LLM comparison and aligns with what SkyPortal will ingest.

Success criteria

- A regex baseline that handles structurally simple cases on a labeled evaluation set.
- An LLM extractor that beats the regex baseline on per-field precision, recall, and F1, with the largest gains on photometry tables and in-prose classification.
- A JSON output schema that conforms to the official GCN schema where possible, with two well-justified additions (SpectralLines, Classification) and an extended Photometry.
- An MCP server exposing extraction-shaped tools (not just search) that SkyPortal or any MCP-aware client can call.
- An evaluation harness that produces a reproducible side-by-side report of regex vs LLM.

2. Existing Repository Assessment

The predecessor repo (sjhend03/GCNMCP) is a TypeScript LeanMCP HTTP server that spawns a Python subprocess per call and routes to a Python backend. It compiles and runs, but the extraction logic is a stub. **Decision (advisor confirmed): start a new repo** with an attribution link to the original in the README. Copy over the reusable Python files (search.py, indexer.py, db.py, utils.py) and the TS LeanMCP bridge; replace the Python extraction backend wholesale.

Component	Location	Verdict
SQLite + FTS5 search	src/search.py	Keep — works, decent ranking
Indexer + upsert pipeline	src/indexer.py, src/db.py	Keep — handles dedup via SHA1
Regex event-name extractor	src/utils.py	Keep — extend with TNS/ZTF/etc.
GCN fetcher (batch)	src/fetch_circulars.py	Keep for backfill, add real-time later
Test suite	tests/	Keep — reuse patterns even if replacing code
LLM extractor (Ollama mistral)	src/tools.py — fetch_and_check_circular_for_grb	Replace — only extracts is_grb + redshift
extract_wavelength stub	src/tools.py:276-305	Delete — empty prompt, undefined model_name var, would crash
Regex GRB checker	src/tools.py — check_for_grb_regex	Replace — subject-line only, single field
TS LeanMCP bridge	leanmcp_bridge/	Keep (advisor confirmed); replace per-call subprocess with long-lived worker

Subprocess overhead note. The current TS layer spawns a fresh Python process per tool call. That was tolerable when calls were "run a regex on a string"; with Claude API calls in the path, you want a long-lived Python worker that the TS layer talks to over a socket or persistent stdin/stdout. Cleaner latency, fewer cold-start surprises, and Python-side caches (HTTP keepalive, model client, prompt cache) actually stick around between requests.

3. Extraction Targets — Optical

Five field categories, per the advisor's specification. Every field is optional at the schema level; the extractor must distinguish "not stated in this circular" from "stated as zero/null".

3.1 Identifiers

- Event/source names: **GRB YYMMDDx**, **AT/SN YYYYxxx**, **TNS/IAU** designations.
- Survey designations: **ZTF**, **ATLAS**, **ASAS-SN**, **Pan-STARRS**, **GOTO**.
- Mission trigger IDs: e.g. Fermi bn230313485, Swift 1159327.
- Cross-references to other circulars: **GCN #NNNNN**, with optional "Circular" word.
- Counterpart-of relations to GW or neutrino events (e.g. "counterpart to GW170817").

3.2 Astrometry

- RA / Dec, sexagesimal or decimal — normalize to decimal degrees, ICRS / J2000.
- Epoch (usually J2000, but record explicitly when stated).
- Position uncertainty: circular radius, or major / minor axis with position angle.
- Localization area at 50% / 90% containment (for GW / neutrino follow-up).

3.3 Time

- Trigger time T0: UTC and MJD.
- Observation start / mid / end timestamps.
- Exposure time (per row in a multi-epoch table).
- Δt -since-trigger phrasings (e.g. "observations began T+234s") — per advisor, extract exactly what the circular says, with units. No cross-circular resolution in v1.

3.4 Photometry

- Telescope and instrument.
- Filter (instrument-specific names allowed: g, r, i, R, clear, etc.).
- Magnitude system: **AB**, **Vega**, or **STMag** (strict enum).
- Magnitude and 1σ uncertainty or flux with units.
- Upper-limit flag and σ level (3σ , 5σ).
- Calibration reference: PS1, SDSS, APASS, 2MASS, Gaia, etc.
- Galactic-extinction-corrected flag (boolean).
- Seeing and airmass when reported.

3.5 Spectroscopy

- Telescope, instrument, grating / grism.

- Wavelength range and spectral resolution.
- Lines identified: rest wavelength, observed wavelength, equivalent width.
- Redshift: value, error, method (spec/photo, abs/em, host/afterglow), confidence.
- Classification + subtype: SN Ia / Ib / Ic / Ic-BL / II / IIn, TDE, AGN, kilonova candidate, etc.
- Phase relative to peak — per advisor, store as free-form string for v1 (often requires external lightcurve context not in the circular).

4. Output Schema Mapping

Most target fields map cleanly to existing schemas in `nasa-gcn/gcn-schema` under `gcn/notices/core/`. Two new schemas are required, and Photometry needs extending. Eventual upstream PR.

Category	Existing schema	Status	Action
Identifiers	Event.schema.json	Sufficient	event_name accepts string or array — drop aliases in directly.
Cross-refs / counterpart	FollowUp.schema.json	Sufficient	ref_type, ref_instrument, ref_ID, reference cover GCN #N and GW/v links.
Astrometry	Localization.schema.json	Sufficient	ra, dec, ra_dec_error (radius or 3-tuple), containment_probability.
Time	DateTime.schema.json	Extend	trigger_time, observation_start/stop, observation_livetime cover absolute times. Per advisor, also extract literal Δt -since-trigger as-is — add small TimeOffset model {value, unit, reference} to capture relative offsets without resolution.
Photometry	Photometry.schema.json	EXTEND (PR)	Advisor confirmed: add telescope, instrument, calibration_reference, galactic_extinction_corrected, seeing, airmass; tighten mag_system to enum [AB, Vega, STMag].
Redshift	Redshift.schema.json	Sufficient	redshift, redshift_error, redshift_measure (spec/photo), redshift_type (em/abs/host).
Reporter	Reporter.schema.json	Sufficient	Use for who issued the alert — do not conflate with photometry telescope/instrument.
Spectral lines	—	NEW	Create SpectralLines.schema.json: array of {rest_wavelength, observed_wavelength, equivalent_width, line_id}.
Classification	—	NEW	Create Classification.schema.json with controlled vocabulary from skyportal/timedomain-taxonomy.

Upstream PR notes

Since these extensions land directly in `Photometry.schema.json`, this is a real PR against `nasa-gcn/gcn-schema`. Two things worth flagging in the PR description: (1) tightening `mag_system` from open string to enum [AB, Vega, STMag] is technically a breaking change for any existing consumer that wrote a non-canonical value — likely fine given how new the schema is, but call it out so reviewers don't have to spot it; (2) the new `SpectralLines` and `Classification` schemas are optical-specific and should sit alongside the existing `Spectral` high-energy schema, not replace it. Naming and file placement worth confirming with the GCN team during review.

Top-level extraction object

Compose all of the above under a single CircularExtraction model that mirrors a circular's contribution to the structured record. Sketch (Pydantic):

```
class CircularExtraction(BaseModel):
    circular_id: int
    event: Event | None = None
    follow_up: FollowUp | None = None
    localization: Localization | None = None
    datetime_: DateTime | None = None
    time_offsets: list[TimeOffset] = [] # literal 'T+234s' captures
    photometry: list[PhotometryExt] = []
    spectroscopy: SpectralLines | None = None
    classification: Classification | None = None
    redshift: Redshift | None = None
    reporter: Reporter | None = None
    extraction_meta: ExtractionMeta # model, prompt version, tokens, latency, $
```

5. Work Plan

Five phases, roughly sequential but with overlap. Each phase produces a concrete, testable artifact.

Phase 1 — Schema, models, and ground truth

- Clone nasa-gcn/gcn-schema; write Pydantic models mirroring Event, FollowUp, Localization, DateTime, Photometry, Redshift, Reporter.
- Draft Pydantic models for new schemas: **SpectralLines, Classification**.
- Draft extended Photometry with telescope, instrument, calibration_reference, galactic_extinction_corrected, seeing, airmass, mag_system enum.
- Add small **TimeOffset** model {value, unit, reference} for literal Δt extractions.
- Compose CircularExtraction top-level model.
- **Pull Vidushi's repo** (nasa-gcn/circulars-nlp-paper): untar data/archive_2025.json.tar.gz for the full backfilled archive, and load tables/topic-modeling-tables/observation_based_topics.csv to subset on Label == "Optical Observations".
- **Verify her labels.** Pass through eval_with_SWIFT/redshift_accuracy.csv to understand which columns are gold-standard (from Swift's manual catalog) vs derived (from her pipeline). Use only the gold ones as ground truth for redshift/GRB#; treat the telescope name column as a useful prior, not a label.
- **Hand-label only the gap.** On a sample of 50 optical circulars she has already topic-labeled, hand-label the ~20 fields she did not cover: full photometry rows (filter, mag, error, system, upper limits, calibration, extinction, seeing, airmass, instrument), RA/Dec + uncertainty, observation times + exposure, classification + subtype, spectroscopy lines, cross-refs to other GCNs, counterpart-of relations. Carry her labels for redshift / GRB# / redshift type forward on those same circulars.
- Stratify the 50: (a) single-row mag report, (b) multi-row mag table, (c) spectroscopic classification with redshift, (d) photometric upper limit, (e) GW/ ν counterpart announcement.
- Write down labeling rules as you go so the dataset is reproducible.

Phase 2 — Regex baseline (parallel with Phase 1)

- Extend src/utils.py event patterns: TNS (AT-prefix YYYY+letters), ZTF, ATLAS, ASAS-SN, Pan-STARRS, GOTO, plus GCN cross-ref pattern.
- Sexagesimal RA/Dec parser → decimal degrees.
- Mag-table detector: identify column-formatted blocks (date, filter, mag, err layouts), parse rows. This is where regex will visibly fail; that is the point.
- Redshift: `z\s*[=~]\s*(\d+\.\d+)` plus nearby-context tags for spec/photo and host/em/abs.

- Classification: walk the timedomain-taxonomy YAML, build alias→canonical map, match in body text.
- Output every extraction in CircularExtraction format so it diffs cleanly against the LLM.

Phase 3 — LLM extractors (Claude + Ollama)

- **Provider abstraction from day one.** Define an Extractor interface; implement ClaudeExtractor (primary) and OllamaExtractor (local comparison). Do not hardcode the client.
- Single prompt, structured output enforced via JSON schema or tool/function calling — not the current "ask for JSON, regex it out of free text" approach.
- Output schema = the Pydantic models from Phase 1.
- Few-shot with 3–5 hand-curated optical circulars covering: multi-row mag table, in-prose classification, upper limit, mag-system inference, GCN cross-ref.
- **Reference Vidushi's prompts** from information-extraction/redshift_extraction.ipynb and figures/Fig6_sample_prompt.pdf before writing your own. Borrow what works; deliberately depart where the new schema demands more. Document the deltas — they will matter in the paper.
- For Claude, run Haiku and Sonnet both initially — Haiku is likely sufficient for structured extraction at much lower cost; measure rather than guess.
- Cache results keyed by (model_name, prompt_version, circular_id, body_hash) so you do not re-pay for the same circular during iteration.
- Handle long-circular case: chunking strategy if body exceeds context, with a deterministic merge.

Phase 4 — Evaluation harness (four-way)

- Four-way comparison on shared input: **regex / Ollama / Claude / Vidushi-baseline**, identical output schema. The Vidushi column comes from her published redshift_accuracy.csv predictions — no re-running her pipeline, just align rows.
- Per-field precision, recall, F1 against the labeled set.
- On Vidushi's four fields (redshift, GRB#, telescope, redshift type), the headline result is **beat her numbers**. On the other ~20 fields, regex vs Ollama vs Claude only (she didn't extract them).
- Numeric fields: tolerance-based correctness (e.g. RA within 0.001°, redshift within 0.001).
- Multi-row tables: row-level matching with set semantics, not list-equality.
- **Cost and latency as first-class metrics**, not afterthoughts: tokens per circular, dollars per 1000 circulars, p50/p95 latency, all reported alongside F1.
- **Cost projection deliverable**: after ~100 circulars through Claude, produce a one-page estimate of (a) one-time backfill cost over the 41k archive and (b) steady-state cost per new circular. Send to advisor.

- Failure-case browser: a small UI or markdown report of cases where one parser beats another.

Phase 5 — MCP server surface + standalone demo

- Keep the TypeScript LeanMCP layer (advisor confirmed). Replace the Python extraction backend wholesale; consider moving from per-call subprocess to a long-lived Python worker the TS layer talks to over a socket — saves cold-start, lets HTTP/model clients persist.
- Replace the stub LLM tool with: **`extract_properties(circular_id)`**, **`get_photometry(event)`**, **`get_classification(event)`**, **`find_counterparts(gw_event_id)`**, **`get_redshift(event)`**.
- Keep search tools (`search_gcn_circulars`, `fetch_gcn_circulars`) — they are still useful.
- Background indexer: when a new circular arrives, run the extractor and persist results to a structured table so MCP queries are cheap.
- **Backfill from Vidushi's archive tarball** (`data/archive_2025.json.tar.gz`) rather than re-scraping. The fetcher only needs to handle May 2025 → present and steady-state going forward.
- Real-time ingestion: poll `gcn.nasa.gov` JSON endpoints for v1 (advisor confirmed). Kafka subscription is a follow-up.
- **Standalone demo deliverable**: a small CLI or chat loop using Claude with the MCP server attached, so the advisor can see end-to-end extraction without needing SkyPortal in the path.

6. Decisions (Resolved)

All blocking questions have been answered by the advisor. Recorded here as the decision log; if any of these change, update both this table and the affected phase in section 5.

#	Decision	Resolution
1	LLM choice	Claude primary, Ollama for comparison. Four-way eval matrix (regex / Ollama / Claude / Vidushi-baseline). Produce a real cost estimate after ~100 circulars.
2	Eval set	Vidushi's labeled set is available (circulars-nlp-paper repo). Use her 13.5k-row redshift dataset directly for those four fields; hand-label 50 stratified optical circulars for the other ~20 fields.
3	Photometry telescope/instrument	Add directly to Photometry.schema.json. Real upstream PR against nasa-gcn/gcn-schema.
4	Δt -since-trigger	Extract literally — absolute or relative, with units, no T0 resolution. Captured via new TimeOffset model.
5	Phase relative to peak	Free-form string for v1.
6	MCP layer	Keep TypeScript (consistency with gcn.nasa.gov stack). Python backend gets rewritten; consider long-lived worker to avoid per-call subprocess cost.
7	Real-time ingestion	Polling JSON endpoints for v1. Kafka is a follow-up.
8	Scope	MCP server only. Must be demonstrable standalone — plan a small demo client. SkyPortal integration is post-v1.
9	Fork vs new repo	New repo with attribution to sjhend03/GCNMCP in the README. Copy the reusable Python files and the TS layer; do not inherit the existing tests.
10	Eval data	Use Vidushi's nasa-gcn/circulars-nlp-paper repo. Provides full archive, topic labels, and 13.5k-row redshift eval set. Hand-label only the gap fields.

Still open (non-blocking). Timeline / hard deadline not specified. Treat the phase ordering as the priority and revisit scope if a deadline emerges.

7. Immediate Next Steps (this week)

With decisions resolved, work begins immediately:

- **Clone Vidushi's repo**; untar the archive; load `observation_based_topics.csv`; filter to optical; cache a working subset of ~500 circulars for iteration.
- Open `information-extraction/redshift_extraction.ipynb` and read it end to end. Note her exact prompt, her validation methodology against Swift, and her reported numbers — they are the baseline to beat.
- Read Sharma et al. 2025 (arXiv:2511.14858) for the published evaluation framing.
- **Create a new repo** (e.g. `gcn-extraction`). Copy over `search.py`, `indexer.py`, `db.py`, `utils.py`, and the TS LeanMCP bridge from `sjhend03/GCNMCP`, with an attribution commit and a README link to the original. Do not fork.
- Stand up the new project: `pyproject.toml`, `ruff`, `pytest`, `pydantic v2`, anthropic SDK, ollama client.
- Sketch the full CircularExtraction Pydantic models (Phase 1, items 1–5).
- **Hand-label 5 circulars** on the gap fields (everything Vidushi did not extract) to a draft labeling spec. This surfaces edge cases early and stress-tests the schema before scaling to 50.
- Wire up the simplest possible ClaudeExtractor on those 5 to validate structured-output ergonomics before building out the full prompt.

8. References

- **Vidushi's data + notebook** — github.com/nasa-gcn/circulars-nlp-paper
- Predecessor repo (attribution) — github.com/sjhend03/GCNMCP
- Official GCN schemas — github.com/nasa-gcn/gcn-schema
- Time-domain taxonomy — github.com/skyportal/timedomain-taxonomy
- Background paper — arxiv.org/abs/2511.14858 (Sharma et al. 2025)
- GCN Circulars archive — gcn.nasa.gov/circulars
- SkyPortal — github.com/skyportal/skyportal
- Python MCP SDK (reference, not in use) — github.com/modelcontextprotocol/python-sdk