ChatGPT

Tools and Libraries for GEO Metadata Access

- **Entrez/E-utilities (Entrezpy, Biopython)**: Libraries to query NCBI's Entrez databases (including GEO) via the E-Utilities API. For example, Entrezpy is a Python package automating complex Entrez queries, caching results, and handling workflows [1]. Entrezpy (and Biopython's Entrez) allow keyword searches (e.g. using disease/tissue terms) and retrieval of XML/JSON records from GEO/SRA, which can then be parsed for metadata. *Capabilities:* flexible search across all GEO fields, integration into Python pipelines [1]. *Limitations:* relatively low-level (users must construct queries and parse responses), and rate-limited by NCBI; no built-in summarization of results.

- **GEOparse (Python)**: A Python library inspired by Bioconductor's GEOquery, designed to fetch and parse GEO records (GSE/GSM/GP L/etc) and their associated data [2]. GEOparse can download a GSE series or GSM sample in SOFT or MINiML format and load it into pandas DataFrames. *Capabilities:* easy retrieval of complete GEO series or sample data, returning structured tables of expression values and metadata [2]. *Limitations:* it works by accession ID and does not perform free-text searches; mainly oriented to microarray/RNA-seq data in GEO, and does not summarize across multiple series or extract high-level statistics.

- **GEOmetadb (R Bioconductor)**: An R package providing a locally downloadable SQLite database of GEO metadata [3]. It "parses all GEO metadata into a SQLite database" that can be queried with SQL or via the R API [3]. *Capabilities:* extremely fast, ad-hoc queries across all GEO fields (e.g. find all GSE with a given tissue or platform); regularly updated database of GEO/SRA entries [3]. *Limitations:* R-centric (requires R/Bioconductor environment) and mostly suited for experienced users comfortable with SQL; not directly a Python or web service.

- **pysradb (Python)**: A Python package/CLI for retrieving metadata and data from SRA and for mapping GEO and SRA accessions [4] [5]. Pysradb can translate between GEO series (GSE) and SRA projects (SRP) or sample/run IDs, and its `metadata` command outputs comprehensive tables (organism, library strategy, run accessions, etc.) for a given SRP [5]. *Capabilities:* tight integration of GEO and SRA (e.g. `gse-to-srp`, `gsm-to-srr`), detailed metadata tables, and raw-data download support. *Limitations:* more focused on sequencing experiments (RNA-seq, ChIP-seq, etc.) than array data; requires knowing at least one accession to start (no natural-language query interface); summary statistics (beyond counts of samples/runs) must be computed separately.

- **GEOfetch (Python/CLI)**: A tool to download and standardize data and metadata from GEO and SRA into a portable format (PEP – Portable Encapsulated Project) [6] [7]. GEOfetch can retrieve GEO series (GSE) or SRA projects by accession or even search terms, and produces unified metadata tables with consistent column names [7]. *Capabilities:* handles both metadata and files (FastQ/BAM) download; merges samples from different projects; can filter by file type/size; provides a unified, "analysis-ready" project structure [7]. *Limitations:* primarily an accession-driven tool (the user must supply GSE/SRP or terms via CLI); requires installation (Bioconda/PyPI); some learning curve to use the PEP format.

- **Geofetch (PEPkit)**: Often conflated with GEOfetch above, Geofetch is essentially the same project (PEPkit's `geofetch` package) described in the literature [6] [7]. It emphasizes compatibility with workflow tools via PEP.

- **OmicIDX Parsers (Python)**: A set of libraries by NCI that parse public genomics metadata into a searchable index. OmicIDX provides parsers for GEO, SRA and others, and can serve queries via an API. *Capabilities:* abstracts away direct API calls by offering a uniform query interface; intended to enable metadata search across resources. *Limitations:* still under active development and not a plug-and-play end-user tool; documentation is limited and it may not cover all metadata fields of interest.

- **Bio.Entrez (Biopython)**: Biopython's wrapper around Entrez E-utilities. It can perform searches (`esearch`) and fetch records (`efetch`) for GEO accessions. *Capabilities:* part of Biopython ecosystem, easy to integrate; *Limitations:* does not manage large queries or automate multi-step queries as Entrezpy does [8].

AI/LLM-Based Metadata Extraction Agents

- **ChIP-GPT (Llama-based)**: A recently published system (ChIP-GPT) fine-tunes a Llama model to extract metadata from SRA/ENA records of chromatin immunoprecipitation experiments [9]. It formulates specific questions (e.g. "what is the ChIP target?") and achieves ~90–94% accuracy in identifying targets and cell lines, even with messy or inconsistent records [9]. *Capabilities:* robust handling of textual metadata with typos or missing labels; requires only modest training examples (100+). *Limitations:* currently specialized to ChIP-seq metadata (the model learned that context); extending to other data types (e.g. WGBS, ATAC-seq) would require additional fine-tuning; depends on having text records as input.

- **Multi-Agent Curation Systems**: Companies like Elucidata have developed multi-agent AI workflows (combining LLMs and specialized sub-agents) to curate complex biomedical metadata. For example, a recent preprint reports an "agentic AI" pipeline that achieves expert-level (>90%) accuracy extracting 23 fields (tissue, disease, etc.) from GEO datasets. This system uses LLM agents for document parsing, ontology mapping (to map terms like "cancer" to disease codes), and inference [10] [11]. *Capabilities:* very high recall on messy GEO metadata; covers diverse metadata fields, combining database entries and related publications. *Limitations:* such systems are generally proprietary and not open-source; require significant engineering to orchestrate multiple agents; also assume use of large LLMs (which may have inference cost or data usage limitations).

- **General LLM with Retrieval (ChatGPT/GPT-4)**: While not a dedicated tool, one could use a general language model with a retrieval step. For instance, a prompt could be crafted to instruct an LLM to query GEO ("use NCBI's database") and return structured info. However, LLMs like ChatGPT cannot directly fetch external data without plugins. A practical approach is Retrieval-Augmented Generation (RAG): retrieve GEO/SRA records via an API or database, then feed the content (or a summary) to the LLM for answering. *Capabilities:* flexible, can handle free-text queries and summarize results in natural language. *Limitations:* requires careful prompt design and possibly vector-indexing of metadata; risk of hallucinations if the retrieval step is weak; currently no off-the-shelf plugin for GEO (unlike e.g. the ScholarAI for papers [12]).

Capabilities vs. Limitations

- **Entrez/E-Utilities**: Can perform arbitrary keyword searches of GEO metadata (e.g. `"disease:brain AND platform:WGBS"`), retrieve matching accessions, and fetch detailed entries [1]. Outputs are machine-readable (XML/JSON). However, interpreting which fields correspond to “tissue” or “disease” requires domain knowledge; Entrez queries can be complex to construct for nuanced prompts. Also, throughput is rate-limited by NCBI.

- **GEOparse/GEOquery**: Very good at loading complete experiments once the ID is known, and handling biological data (expression matrices, etc.). But they assume the user knows which series to download. They do not help in *finding* relevant studies from a natural-language question – this must be done via Entrez or keyword filtering.

- **GEOmetadb**: Excels at answering structured queries over GEO metadata (e.g. SQL queries for all human brain WGBS datasets). It is perhaps the gold standard for metadata retrieval. However, it is R-only and requires keeping the SQLite DB updated. It also returns raw fields (e.g. series titles or summary texts) rather than semantic labels; mapping a prompt (“cancer”) to actual GEO “disease” fields requires understanding conventions or using associated vocabularies (e.g. MeSH, DOID).

- **pysradb**: Very useful for linking between GEO and SRA, and pulling summary tables of sequencing projects (numbers of runs, read counts, etc. as shown in its output [5]). It could help in summarizing a dataset’s size or technology. But pysradb focuses on the underlying SRA data; it won’t itself interpret natural language (e.g. it has no “search by tissue” feature beyond what Entrez offers). Its metadata fields are also mostly technical (instrument, layout, etc.), so it may not provide fields like “disease” unless they appear in the title/description.

- **GEOfetch**: Combines many strengths – it can use NCBI search under the hood and fetches data and metadata in one step, yielding a project structure ready for analysis [6] [7]. For example, one could script `geofetch WGBS human brain cancer` to retrieve matching series and get a table of samples and raw files. It normalizes many metadata columns. Its limitations are mainly practical: it expects command-line use (though it has a Python API) and may be overkill if one only needs a quick lookup.

- **LLM Agents (ChIP-GPT, etc.)**: Can extract nuanced fields from unstructured text (e.g. identify that “hippocampus” is brain tissue, or that “glioblastoma” is a type of cancer) even if the record format is inconsistent. They effectively add “domain reasoning” on top of raw metadata. The trade-off is that they require substantial training data or few-shot engineering, and they depend on the quality of source text. If the metadata are sparse, no LLM can create data out of thin air – it can only reason about what is present.

Proposed System Architecture

To implement a GEO metadata/summary agent (answering prompts like “metadata of WGBS human brain cancer data”), we recommend extending existing tools rather than starting entirely from scratch. A high-level architecture might be:

1. **NLP Query Interface** – A front-end (e.g. a chat or web UI) where the user’s natural-language request is received. A small prompt-understanding module (possibly an LLM) parses the request into keywords and filters (e.g. “disease=cancer”, “tissue=brain”, “assay=WGBS”, “organism=human”). Using ontology lookup (e.g. mapping “WGBS” → “Bisulfite-Seq” or linking “cancer” to disease ontology IDs) can improve precision.

2. **Query Builder** – The parsed filters are translated into database/API queries. For example, an Entrezpy agent or a direct SQL on GEOmetadb could be used: e.g. `db=geo&term="("cancer"[Organism] AND "brain"[Title]) AND WGS[STRATEGY]"`. Alternatively, one could use GEOfetch’s search mode to retrieve matching GSE/SRP IDs.

3. **Data/Metadata Retrieval** – Using the accession list from step 2, employ tools to gather data. For example:

4. **Metadata Extraction**: Use GEOfetch or pysradb to download sample/run metadata for each study. This yields tables of attributes (sample title, platform, library strategy, etc.).

5. **Summary Statistics**: If required, compute simple stats (e.g. number of samples, total bases) using pysradb outputs or by programmatic calls (e.g. count runs or reads from SRA records). For expression data, one might use GEOparse to get expression matrices and then compute gene count summaries.

6. **Aggregation and Structuring** – Consolidate the results into a structured format (Pandas DataFrames or JSON). For instance, one table might list all matching GSE IDs with titles and sample counts; another lists key metadata fields for those studies (platform, sample demographics, etc.). Ensure consistent column names (as GEOfetch does).

7. **LLM Summarization/Answer Generation** – Feed the structured data (or key excerpts) to an LLM (or a templated generator) to produce a concise answer. The LLM can highlight the top studies, mention the range of values (e.g. “There are 5 WGBS studies of human brain cancer in GEO, with a total of 200 samples”), and answer any sub-questions. If needed, fine-tune the LLM on example Q&A pairs (as ChIP-GPT did) or use few-shot prompts to get domain-appropriate language.

8. **Output Formatting** – Return both the human-readable summary and the raw data. For example, the system can output JSON or CSV tables (suitable for downstream analysis) along with a text snippet summarizing findings. The use of a PEP (Portable Encapsulated Project) format (as GEOfetch does) would allow easy pipeline integration.

9. **Multi-Agent Orchestration (optional)** – For complex queries, multiple sub-agents could operate: e.g. a “retriever agent” runs the Entrez/SQL queries; a “parser agent” cleans up returned metadata;

an “ontology agent” standardizes terms. An orchestrator (like in the Elucidata system) schedules these tasks. While ambitious, even a simple synchronous pipeline (steps 1–6 in sequence) can work.

Innovation: This design combines robust bioinformatics tools (Entrezpy, GEOmetadb, pysradb/GEOfetch) with modern NLP/AI. The traditional tools ensure accuracy and completeness of metadata retrieval, while the LLM layer provides flexible interpretation of user language and summarization. By reusing existing metadata indices (GEOmetadb) and wrappers, we avoid rebuilding the entire GEO index from scratch. Instead, we “wrap” these tools in an intelligent interface. The LLM (like GPT-4 or an open alternative) could even be used for query construction and result interpretation, effectively bridging natural-language and database search. All output (tables of metadata, accession lists, counts) can be formatted in JSON or pandas-friendly form for downstream ML pipelines.

---

[1] [8] Entrezpy: a Python library to dynamically interact with the NCBI Entrez databases - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC6821292/

[2] GEOparse Documentation
https://geoparse.readthedocs.io/_/downloads/en/latest/pdf/

[3] Bioconductor - GEOmetadb
https://www.bioconductor.org/packages/release/bioc/html/GEOmetadb.html

[4] [5] GitHub - saketkc/pysradb: Package for fetching metadata and downloading data from SRA/ENA/GEO
https://github.com/saketkc/pysradb

[6] [7] GitHub - pepkit/geofetch: Builds a PEP from SRA or GEO accessions
https://github.com/pepkit/geofetch

[9] (PDF) ChIP-GPT: a managed large language model for robust data extraction from biomedical database records
https://www.researchgate.net/publication/377979123_ChIP-GPT_a_managed_large_language_model_for_robust_data_extraction_from_biomedical_database_records

[10] So excited about the launch of our multi-agentic AI system. | Milan Gupta
https://www.linkedin.com/posts/milan-gupta4_multi-agent-ai-system-for-high-quality-metadata-activity-7338619264934817793-pmRS

[11] Interested in high quality curation at scale? | Krutika Satish Gaonkar
https://www.linkedin.com/posts/krutika-satish-gaonkar-517a5535_interested-in-high-quality-curation-at-scale-activity-7339027080648560642-eLXt

[12] Is there a way to get chat gpt to access https://www.ncbi.nlm.nih.gov?
https://www.reddit.com/r/ChatGPT/comments/13izz5p/is_there_a_way_to_get_chat_gpt_to_access/