Synoptic Key of Life Pipelines

by La Monte Henry Piggy Yarroll <piggy.yarroll@gmail.com>

Synopic Key of Life (SKOL) is a project to understand the open taxonomic literature with machine learning. A Synoptic Key is a tool that a biologist can use to search for formal descriptions that match the organism they have in front of them.

The goal of the project is to support amateurs in producing good formal descriptions, leading them closer to identification of their collections.

SKOL Background and State of the Art

The principal researcher on this project is most familiar with the Mycological (Fungi) literature in English, so that is the initial target.

Web scrapers

The existing web scrapers are shell scripts based on `wget` that extract issues and articles from archives for the literature. The current output of the web scraping stage is directory hierarchies of text files, largely extracted directly from PDFs. Some are hand-built hierarchies that have had images extracted from the PDFs with `tesseract` OCR applied to them.

Taxon name and description extraction

There is an existing PySpark pipeline which takes the raw text and extracts taxon names (mostly species) and their matching descriptions with a machine learning classifier. The classifier produces annotated files for ingestion by the next stage.

Dr. Drafts and the SBERT embeddings

The heart of the SKOL is a set of sentence embeddings of the species descriptions used to find descriptions that are similar to the description being produced by the user of the synoptic key. Dr. Drafts ingests a directory full of files to build the SBERT embeddings.

JSON embeddings for User Interface

There is an existing pipeline based on the Mistral Small Language Model (SLM) which converts prose descriptions into JSON structures which are features, subfeatures, and their values. The current tool just produces JSON files.

The purpose of the JSON embeddings is to produce feature suggestions for use while building descriptions. The idea is that small human-interpretable decision trees can be produced from the top few matching descriptions to guide the user in their selection of features to investigate.

Proposed extensions

There are four areas which could benefit from Big Data methods: Ingestion and OCR, Description Extraction, Ingestion into Dr. Drafts, and storage of the UI JSON embeddings. There is an overall need for tracking the metadata for each journal issue, article, and description.

Depending on the availability of collaborators, two or more of these areas could be solved for this class.

Metadata tracking

It is necessary to track provenance for data carefully. Items in the metadata include the name of the journal, Bibtex entries, human and download URLs, any internal keys, and the date of the scrape. A conventional RDBMS is probably sufficient for this data, as the scale of these entries is probably not in the big data range.

Ingestion and OCR

Most journals make their material available via PDF. Older material is often distributed as PDFs of page images, and OCR is needed. Because of the cost of OCR, it makes sense to store the text version of the documents.

A document store like Hadoop or Minio makes sense for both the retrieved files, and the extracted text.

Description extraction

The species name and description extraction code needs to be updated to pull from the text repository. The descriptions are on the order of 1K to 2K, so are very small documents. There is additional metadata generated in this stage which includes page numbers (physical and logical), and keys linking back to the Ingestion metadata.

Again, a document store is probably the right medium for the taxon name and description data. A format like parquet is probably a good choice for the resting form.

Dr. Drafts ingestion

The Dr. Drafts system currently imports its data from files on the filesystem. Each data source in Dr. Drafts has its own ingestion code, so it should be practical to add an adapter to pull directly from the Description extraction resting state.

Currently, Dr. Drafts is single threaded, but it should be possible to add a PySpark core that computes embeddings.

The embeddings are stored as a large Python pickle on the filesystem. It may make sense to store embeddings in a distributed structure.

JSON UI records

The JSON embeddings generated by Mistral are currently calculated and stored in files on the filesystem. This appears to be an ideal application for CouchDB to store the generated JSON and linking metadata.

Cladogram data

More investigation is needed for building cladograms, but it appears that a graph database would be a good intermediate form for the descriptions used for making cladograms.

Research Questions

The core research question addressed by SKOL is “Which literature has descriptions similar to my collection?”

Two new questions which could be addressed are

- Are there cryptic synonyms in the literature?
- Can we build a morphological cladogram for all of Mycology?

Cryptic synonyms are pairs of descriptions in the literature which overlap significantly. Running clustering on the SBERT embeddings could be used to find such pairs.

A cladogram is a diagram that shows distances between taxa in a hierarchy. This can also be built by examining clusters in the SBERT embeddings. Graph databases are often used when building cladograms.

Division of Labor

The amount of work that is in scope for this project depends on success with recruiting fellow students to the project. There are clearly components that can be pulled out as separate work items.