Project Description

Project:

Automation of Paper Entry into ORKG / Neo4j

Delivery:

A Python library containing a set of functions that automate the paper-filling on ORKG.

1. Background and Context

This project aims to automate the process of adding structured research data to the Open Research Knowledge Graph (ORKG), which might in the future also integrate with Neo4j for knowledge representation and querying.

The source data originates from a coding scheme developed through manual annotation of research papers by two different coders. This dataset is currently stored in an Excel spreadsheet, where each row represents a paper along with metadata and extracted content.

Manually entering this information into ORKG is time-consuming and inefficient. Automating this process will save time, reduce human error, and enable large-scale integration of scholarly knowledge.

2. Objectives

- Automate the ingestion of papers and metadata into ORKG using predefined templates.
- Ensure compatibility with the ORKG API.
- Maintain flexibility for both sandbox and production environments.
- Structure data in a format that is compliant with ORKG's data model.

3. System Requirements

Functional Requirements

- Load structured data from Excel files containing coded research paper information.
- Connect to the ORKG platform (sandbox or production) using secure credentials.

- Process each paper entry to match the format expected by the ORKG template.
- Submit each processed paper as a new resource to ORKG.
- Return the ORKG resource URL for each successfully added paper.

Non-Functional Requirements

- Maintaining code documentation standards and clean code (Use of unit testing is not required, but always welcomed).
- Use of reliable and well-documented libraries (e.g., numpy, NLTK).
- Error handling and logging for connection issues, submission failures, or malformed data.
- Log File: Write every step into a log file for review and debugging purposes, with an option to turn this feature on and off.
- The produced Code should be modular and reusable for future extensions (e.g., comparison generation, Neo4j sync, or connecting to other Knowledge Graph libraries).

4. Required Packages / Dependencies

- ORKG Python Client
- pandas

5. Function Definitions (Suggested)

read_excel_file(file_path: str) -> pd.DataFrame

Reads an Excel file and returns its contents as a pandas DataFrame.

connect_to_orkg(host: str, credentials: Tuple[str, str]) -> ORKG

Establishes a connection to the ORKG platform (production or sandbox).

connect_to_template(template_id: str) -> bool

Retrieves and caches a specified ORKG template by its resource ID.

process_all(Papers DataFrame) -> bool

Iterate through the rows and call the function process_paper for each row

process_paper(paper_row: pd.Series) -> dict

Transforms a single row of the Excel DataFrame into a structured dictionary matching the ORKG format.

add_paper(paper_data: dict) -> str

Submits a structured paper dictionary to ORKG and returns the resource URL.

6. Optional Extensions

- Comparison Generation: Automatically compare papers within the same template class.
- Neo4j Export: Export extracted and structured paper data into a Neo4j database.
- Data Validation Module: Ensure the integrity and consistency of data before submission.
- Visualize the paper graph.

7. Important Resources

- ORKG Python Documentation:

https://orkg.readthedocs.io/en/latest/introduction.html

- ORKG Sandbox environment:

https://sandbox.orkg.org

- ORKG Contribution Template [Class ID: C113016]:

https://sandbox.orkg.org/template/R879140

- Example paper on ORKG:

https://sandbox.orkg.org/paper/R879151

- Neo4J Documentation:

https://neo4j.com/docs/