

## 1 Objectives

Students work in teams to tackle the three tasks of Data Collection, Data Management, and Data Analysis that you have learnt in the lecture classes, applying the concepts of data quality assessment and exploratory data analysis (EDA) introduced earlier. Concretely, you will:

- Collect structured metadata for a subset of arXiv papers via the Semantic Scholar API.
- Store the raw JSON responses into a MongoDB Atlas collection.
- Transform the JSON documents into a tabular CSV file.
- Analyze the data quality of the resulting table using several relevant aspects discussed in the data quality lecture.
- Perform EDA to extract and justify non-trivial insights about the scraped papers.

You are expected to coordinate within the team and divide work efficiently.

## Task 1: Semantic Scholar Metadata Collection and MongoDB Storage

Recalling the range of IDs from arXiv assigned to your team in the laboratory milestone, you will work with this range for this exam.

### 1.1 Range IDs

Your student IDs serve as a basis to determine which articles you will work on in this assignment. Define

- $S$ : the sum of the numerical values of all team members' student IDs.
- $k = S \bmod 50$ : an integer in  $\{0, 1, \dots, 49\}$ .

Each arXiv identifier has the form  $yy\text{mm}.\text{xxxxx}$ , where the *right part* is the 5-digit suffix after the dot. For every arXiv ID in the range assigned to your team, extract the right part as an integer  $r$  and select the ID if

$$r \equiv k \pmod{50}.$$

The set of selected IDs will be the target papers for this exam.

### 1.2 Semantic Scholar API

For each selected arXiv ID, use the Semantic Scholar Graph API endpoint for arXiv papers. Your code should:

1. Construct the appropriate API URL for a paper identified by its arXiv ID.

2. Request all fields needed later for the CSV schema in Task 2, including metadata such as identifiers, title, abstract, venue, year, citation statistics, fields of study, publication information, journal information, authors, and embeddings.
3. Respect the API rate limits (for example, by inserting a suitable delay between requests or implementing a simple rate-limiting mechanism).
4. Handle missing or partial responses gracefully (for example, retries with a capped number of attempts, or logging and skipping entries that repeatedly fail).
5. Log the list of successfully collected arXiv IDs and any IDs that could not be retrieved, together with short error messages.

### 1.3 MongoDB Atlas Collection

Use the MongoDB Atlas cluster and database that your team has already set up in the previous laboratory sessions.

- Create (or reuse) a collection dedicated to this exam, for example `semanticScholarPapers`.
- Insert one JSON document per successfully retrieved paper. The document should correspond closely to the original Semantic Scholar JSON response, without flattening or dropping fields at this stage.
- Ensure that the arXiv ID and Semantic Scholar `paperId` are stored explicitly so that each document can be uniquely identified and re-queried.
- After insertion, verify (via a simple query) that the number of documents in the collection matches the number of successfully retrieved papers.

Your code for this task should be fully automated: given the assigned arXiv range and your team members' student IDs, it must be able to compute  $k$ , determine the target arXiv IDs, call the API, and populate the MongoDB collection.

## Task 2: JSON-to-CSV Transformation and Data Quality Analysis

### 1.4 Building the CSV File

Starting from the MongoDB collection created in Task 1, write Python code that:

1. Connects to MongoDB Atlas and retrieves all documents belonging to this exam.
2. Parses each JSON document and extracts or computes the fields listed below.
3. Constructs a tabular dataset and exports it to a CSV file.

Each row in the CSV file must contain at least the following columns, in this exact order:

- `paperId`
- `corpusId`

- title\_length
- title\_length\_without\_space
- title\_length\_in\_words
- abstract\_length
- abstract\_length\_without\_space
- abstract\_length\_in\_words
- venue\_length
- venue\_length\_without\_space
- venue\_length\_in\_words
- venue
- year
- referenceCount
- citationCount
- influentialCitationCount
- isOpenAccess
- fieldsOfStudy
- s2FieldsOfStudy\_count
- publicationTypes\_count
- publicationDate
- journal\_name
- author\_count
- first\_author\_name
- embedding\_dimension
- embedding\_mean
- embedding\_stddev

Document clearly in your code how each column is derived from the JSON structure. For example:

- title\_length: number of characters in the paper title (including spaces).
- title\_length\_without\_space: number of characters in the paper title, excluding whitespace.
- title\_length\_in\_words: number of tokens obtained by splitting the title into words.

- `abstract_length`, `abstract_length_without_space`, `abstract_length_in_words`: defined analogously for the abstract.
- `venue_length`, `venue_length_without_space`, `venue_length_in_words`: defined analogously for the venue string; handle empty or missing venues carefully.
- `fieldsOfStudy`: a list-valued column containing the fields of study reported by Semantic Scholar (if any).
- `s2FieldsOfStudy_count`: length of the corresponding Semantic Scholar fields-of-study array.
- `publicationTypes_count`: number of publication types associated with the paper.
- `journal_name`: journal or conference name, if available; otherwise leave empty or use a clear missing-value indicator.
- `author_count`: number of authors.
- `first_author_name`: name of the first author in the author list.
- `embedding_dimension`: length of the embedding vector returned by the API; if no embedding is present, this value should indicate missingness.
- `embedding_mean` and `embedding_stddev`: mean and standard deviation of the embedding vector entries, when available.

The resulted CSV file should be stored as `papers.csv`.

## 1.5 Data Quality Assessment

After you have built the CSV file, perform a concise yet meaningful analysis of its data quality. In this part, you should:

- Briefly recall, in your own words, the notion of data quality as introduced in the lectures.
- Choose multiple relevant data quality aspects from those discussed in class (for example, issues related to correctness, completeness, consistency, timeliness, interpretability, etc.), covering more than just one or two dimensions.
- For each chosen aspect, define how you operationalize it in this context (that is, how you measure or detect problems using the columns in your CSV file).
- Provide quantitative summaries or simple visualizations that support your assessment (for example, counts of missing values, distribution of suspicious entries, proportion of rows affected by a certain issue).
- Discuss the main data quality issues you found, their potential impact on downstream analysis or modeling, and simple strategies you would recommend to mitigate them.

The aim of this part is not to apply every concept from the data quality lecture, but to show that you can select, justify, and apply several suitable notions to a concrete dataset.

## Task 3: Exploratory Data Analysis (EDA)

Using the CSV table from Task 2, perform an EDA to discover and articulate valuable insights about the scraped papers. Your EDA should integrate both numerical summaries and visualizations, and it must be guided by questions that you identify yourself.

Begin by exploring the dataset to discover interesting patterns and formulating your own guiding questions. You may think about questions related to citation behaviour, publication years, title and abstract lengths, open-access status, fields of study, or any other variables that appear meaningful from your CSV table. Use the examples and techniques from the lecture slides as inspiration, but adapt them to the particular structure and content of your dataset.

For each question or pattern you investigate, decide which EDA techniques are appropriate for the variables involved. Select numerical summaries and plots that match the nature of the variables (for example, whether they are categorical or numerical, and whether you are examining a single variable or relationships between several variables), and implement them in your notebook.

For each main question you pose and each major pattern you report, provide a short written interpretation that:

- Explains what the visualization or summary statistic shows in plain language.
- Interprets why this pattern might be reasonable, surprising, or useful in the context of scientific papers.
- Acknowledges limitations due to the sample size, the modulo selection rule, or data quality issues identified in Task 2.

Your goal is not only to generate plots, but to turn those plots into convincing and well-argued insights.

**Note:** Each team should have at least **six** significant insights submitted for this task.

## 2 Submission

Each team must submit the following items:

- **Notebook 1 – Task 1 (Data collection and storage):** a Jupyter notebook that computes  $k$ , selects arXiv IDs, calls the Semantic Scholar API, and inserts the raw JSON documents into MongoDB Atlas.
- **Notebook 2 – Task 2 (Transformation and data quality):** a Jupyter notebook that loads data from MongoDB, constructs the CSV file with the required schema, exports it, and carries out the data quality assessment.
- **Notebook 3 – Task 3 (EDA):** a Jupyter notebook that performs the EDA, including all code for summaries and plots.
- **Intermediate CSV file:** the CSV table produced in Task 2 and used as input for Task 3.

Before submission, you must re-run each notebook from top to bottom so that all cells are executed and all results are visible in the output.

Organize your submission as follows:

- Create a single folder named after the student ID of the group's representative.
- Inside this folder, place the three notebooks named exactly `task-1.ipynb`, `task-2.ipynb`, and `task-3.ipynb`, together with the intermediate CSV file.
- Compress this folder into a single ZIP file with the same name as the representative's student ID or `StudentID.zip`.
- Upload this ZIP file to Moodle according to the instructions on the course page.

A visual representation of the expected directory structure is:

```
<StudentID>/
  └── task-1.ipynb
  └── task-2.ipynb
  └── task-3.ipynb
  └── papers.csv
```

Here `StudentID` stands for the student ID of the group's representative, and `papers.csv` denotes the intermediate CSV file produced in Task 2.

All explanations, reasoning, and answers should be written as Markdown cells within the three notebooks. No separate written report is required unless announced otherwise by the instructor, depending on the in-class situation.

### 3 Grading Distribution

The grading is decided for each task with the final mark is weighted with the below distribution:

- Task 1: 30%.
- Task 2: 25%.
- Task 3: 45%.