Working with Scholarly Metadata

> *some thoughts, some tips*

Asura Enkhbayar | April 3rd, 2023

Outline

Outline

- (*some thoughts*) on **scholarly metadata**
    - provenance debt

Outline

- (*some thoughts*) on **scholarly metadata**
    - provenance debt
- (*some tips*) for **working with**
    - data
    - projects
    - code

… Scholarly Metadata

… Scholarly Metadata

What even is scholarly metadata or where to begin with scholarly metadata?

How can *provenance debt* help to answer these questions?

https://docs.google.com/presentation/d/1cQoPxYi49n4GuEHsxayrtj3S74BVRrlqRenYpteAIts
usp=sharing

Working with …

- Data sources, APIs, Clients
- Organizing projects, data, code
- Code stuff

Data sources, APIs, clients

Tip 1 - Read the docs

1. **Read the docs**
2. **Read the docs**
3. **Read the docs**

Tip 1 - Read the docs

1. **Read the docs** of the data source
2. **Read the docs** of the API
3. **Read the docs** of the API client

Example - working with Crossref

Data Source

- Crossref schema library is useful but mostly intended for publishers

REST API

- Current REST API documentation lives at https://api.crossref.org/
- **However**, the deprecated documentation provides many concrete examples and use-cases
- Similarly, Github Issues are a great resource for questions

API Clients

- https://github.com/sckott/habanero
- https://commonmeta-py.docs.front-matter.io/

Tip 2 - Learn `reqests`

API clients are great but learn how to work with APIs directly.

- Recommendation: requests

Data Science Projects

Tip 3 - Think about structure

- Project
    - What kind of outcomes are expected? Research article, software, datasets...
- Data structure
    - Outline the processing pipeline
- Code structure
    - Notebooks, scripts, local processing vs server, ...

The Cookiecutter Data Science Template is great but an overkill for most research projects. Instead, this SCL example might be helpful.

Tip 4 - Document changes

Don't shy away from (re-)organizing files, data, and code as the project evolves.

However, try to document these changes. You will think yourself when writing your methods sections...

I've previously (mis)used Github Wiki pages as a research changelog.

Tip 5 - Managing code environments & dependencies

- Managing Python distributions at system level: `pyenv`
- Installing Python programs like regular apps: `pipx`
- Managing project dependencies & environments: `poetry`

Coding in Python

Tip 6 - Streaming data with JSON Lines

Too small for big data, but still too big for RAM

- Consider using JSON lines over CSV/JSON for raw data
- Sampling and streaming are your friends
- Pandas is amazing but not made for this kind of stuff

Tip 6 - Streaming data with JSON Lines

Too small for big data, but still too big for RAM

- Consider using JSON lines over CSV/JSON for raw data
- Sampling and streaming are your friends
- Pandas is amazing but not made for this kind of stuff

In [ ]:
```python
import pandas as pd

# Reading .jsonl
df = pd.read_json("input.jsonl", lines=True)

# Writing .jsonl
df.to_json("output.jsonl", lines=True, orient="records")
```

Tip 7 - Document your collection process

There is never too much metadata about our collection processes

- ISO timestamps

Tip 7 - Document your collection process

There is never too much metadata about our collection processes

- ISO timestamps

```
In [ ]: import datetime

        ts = datetime.datetime.now().isoformat()
```

Tip 7 - Document your collection process

There is never too much metadata about our collection processes

- ISO timestamps

In [ ]:
```
import datetime

ts = datetime.datetime.now().isoformat()
```

- Handle those exceptions

Tip 7 - Document your collection process

There is never too much metadata about our collection processes

- ISO timestamps

In [ ]:
```
import datetime

ts = datetime.datetime.now().isoformat()
```

- Handle those exceptions

In [ ]:
```
try:
    response = requests.get()
except Exception as e:
    error = e
```

Tip 8 - Don’t overwrite your files...

> *just don't...*

Tip 8 - Don’t overwrite your files...

> *just don't...*

```
In [ ]: from pathlib import Path

        output_file = Path("this/took/a/week/to/process.csv")

        # inelegant lifesaver
        if output_file.exists():
            sys.exit()

        print("you can thank me later")
```

Tip 10 - Nice progress bars are amazing

> *Just start using `tqdm` ❤️*

https://github.com/tqdm/tqdm

Tip 10 - Nice progress bars are amazing

> *Just start using `tqdm`* ❤️

https://github.com/tqdm/tqdm

In [ ]:
```
import time

from tqdm.notebook import tqdm

long_list = list(range(0,50))

for item in tqdm(long_list):
    time.sleep(0.5)
```

One final example

Putting together those tips to query the Crossref API with a list of DOIs

One final example

Putting together those tips to query the Crossref API with a list of DOIs

```
In [ ]: def collect_references(overwrite: False):
            if crossref_responses_f.exists():
                if overwrite:
                    with open(crossref_responses_f, 'w') as f:
                        for doi in tqdm(dois):
                            ts = datetime.datetime.now().isoformat()
                            error = None

                            try:
                                response = cr.works(doi, warn=True)

                                if response['status'] != 'ok':
                                    error = response['status']
                                else:
                                    f.write(json.dumps(response["message"]) + "
                            except Exception as e:
                                error = e

                            # Update collection progress
                            update_collection_progress(doi, ts, error)
                else:
                    print("file already exists")
```