Way Lab: Research in Progress

pycytominer-transform

Presentation Outline

1. ✅ Overview
2. 🛣️ Points of Interest
    - ⚙️ MapReduce influence
    - 📚 Data grammar

Overview • What?

🛠️ `pycytominer-transform` is a solution to data *scalability* and *compatibility* challenges with large cell morphology feature datasets.

Overview • What?

https://cytomining.github.io/pycytominer-transform/

Overview • How?

Quick implementation details:

- 🐍 Python 3.8 - 3.9
- 📝 Prefect workflows
- 🗃️ Arrow core data format
- 🤸 DuckDB data gymnastics

Overview • How?

Quick auxiliary details:

- 🐍 Poetry environments
- ✅ Pre-commit checks
- 🎛️ Dagger continuous testing
- 📖 Myst for docs over `.md`

Overview · Why?

- Cell morphology datasets are large.
- Data transformation is needed.
- `pandas` doesn't scale well.
- CSV's and SQLite don't scale well.

MapReduce Influence

Prefect provides a "map" capability leveraged in workflows for parallelism.

MapReduce Influence

Google Research:

MapReduce: Simplified Data Processing on Large Clusters

"MapReduce is a programming model and an associated implementation for processing and generating large data sets."

MapReduce Influence

MapReduce Influence

`pycytominer-transform` leverages multiple maps to parallelize data extract, transform, and reduction (accumulation).

Data Grammar

What is a data "join"?

Data Grammar?

Apache Arrow / ARROW-18246

[Python][Docs] PyArrow table join docstring typos for left and right suffix arguments

Details

Type:	Bug	Status:	RESOLVED
Priority:	Minor	Resolution:	Fixed
Affects Version/s:	None	Fix Version/s:	11.0.0
Component/s:	Documentation		
Labels:	docs-impacting documentation pull-request-available		
External issue URL:	https://github.com/apache/arrow/issues/33429		

Link: ARROW-18246

How do we talk about data pro*gramm[ar]ing*?

Data Grammar

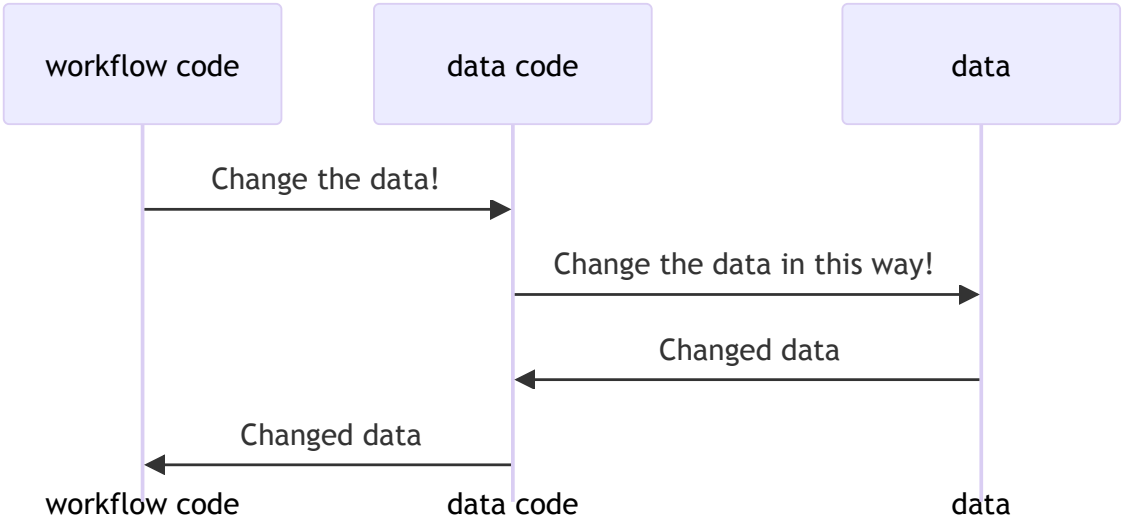
Have we done a good job protecting
against software decay for data code?

Data Grammar • Some Translations

`words` + `syntax` = `meaning`

`data` + `instructions` = `result`

Data Grammar • Layers



Data Grammar • Examples

Example input

id	data1
1	a
2	b
3	c

id	data2
1	d
2	e
3	f

Data Grammar • Examples

Example output

id	data1	data2
1	a	d
2	b	e
3	c	f

Data Grammar • Examples

```
# native python "join"
dataset_a = {"id":[1, 2, 3], "data1": ["a","b","c"]}
dataset_b = {"id":[1, 2, 3], "data2": ["d","e","f"]}

dataset_c = dataset_a.copy()
dataset_c.update(dataset_b)

print(dataset_c)
# {
#   'id': [1, 2, 3],
#   'data1': ['a', 'b', 'c'],
#   'data2': ['d', 'e', 'f']
# }
```

Data Grammar • Examples

```
# pandas merge to achieve a join
import pandas as pd

dataset_a = pd.DataFrame(
    {"id": [1, 2, 3], "data1": ["a", "b", "c"]},
)
dataset_b = pd.DataFrame(
    {"id": [1, 2, 3], "data2": ["d", "e", "f"]},
)

dataset_a.merge(
    dataset_b,
    on="id",
```

Data Grammar • Examples

```
# pandas merge
import pandas as pd

dataset_a = pd.DataFrame(
    {"id": [1, 2, 3], "data1": ["a", "b", "c"]},
)
dataset_b = pd.DataFrame(
    {"id": [1, 2, 3], "data2": ["d", "e", "f"]},
)

dataset_a.merge(
    dataset_b,
    on="id",
```

Data Grammar • Examples

```
# pyarrow join
import pyarrow as pa

dataset_a = pa.Table.from_pydict(
    {"id": [1, 2, 3], "data1": ["a", "b", "c"]},
)
dataset_b = pa.Table.from_pydict(
    {"id": [1, 2, 3], "data2": ["d", "e", "f"]},
)

dataset_a.join(
    dataset_b,
    keys="id",
```

Data Grammar · Decay

- We've repeated ourselves *and*
- Used inconsistent keywords *and*
- May witness different results.

Data Grammar • SQL

SQL was created 49 years ago to enable code-based data operations.

SQL provides one way to isolate and understand data grammar.

Data Grammar • DuckDB SQL

`Arrow + DuckDB SQL = result`

Data Grammar • DuckDB SQL

```
import duckdb
import pandas as pd

dataset_a = pd.DataFrame(
    {"id": [1, 2, 3], "data1": ["a", "b", "c"]},
)
dataset_b = pd.DataFrame(
    {"id": [1, 2, 3], "data2": ["d", "e", "f"]},
)

joined_result = (
    duckdb.connect()
    .execute(
        f"""
    SELECT
        dataset_a.id,
        dataset_a.data1,
        dataset_b.data2,
    FROM dataset a
```

Data Grammar • DuckDB SQL

```
import duckdb
import pyarrow as pa

dataset_a = pa.Table.from_pydict(
    {"id": [1, 2, 3], "data1": ["a", "b", "c"]},
)
dataset_b = pa.Table.from_pydict(
    {"id": [1, 2, 3], "data2": ["d", "e", "f"]},
)

joined_result = (
    duckdb.connect()
    .execute(
        f"""
    SELECT
        dataset_a.id,
        dataset_a.data1,
        dataset_b.data2,
    FROM dataset a
```

Data Grammar • DuckDB SQL

Creating data grammar with SQL using Arrow and DuckDB creates understandable and performant data code.

Thank you

Questions / comments?