

Exercise: Identity Resolution

Web Data Integration



The PyDI Framework

- The **Python Data Integration Framework** (PyDI) provides methods for end-to-end data integration
- Implements methods for
 - Data Pre-Processing
 - Schema Matching
 - Identity Resolution
 - Data Fusion
 - Evaluation
- Open Source under Apache 2.0 License
- <https://github.com/wbsg-uni-mannheim/PyDI>

PyDI Tutorial

https://github.com/wbsg-unimannheim/PyDI/blob/main/docs/tutorial/entity_matching_and_fusion/data_integration_tutorial.ipynb

PyDI Data Integration Tutorial

This tutorial demonstrates comprehensive data integration using PyDI. We'll work with movie datasets to showcase the data integration pipeline from entity matching to Data Fusion.

What You'll Learn

1. **Data Loading & Profiling:** Load and analyze movie datasets with provenance tracking
2. **Entity Matching:**
 - Blocking strategies (Standard, Sorted Neighbourhood, Token-based, Embedding-based)
 - Multi-attribute similarity matching with custom comparators
 - Machine learning-based entity matching
3. **Data Fusion:**
 - Conflict resolution with custom fusion rules
 - Quality assessment against test set
 - Provenance-based conflict resolution

Datasets

We'll use three movie datasets:

- **Academy Awards:** Movies with Oscar information (4,592 records)
- **Actors:** Movies with actor details (149 records)
- **Golden Globes:** Movies with Golden Globe awards (2,286 records)

These datasets contain overlapping movie information but with different attributes, data quality issues, and conflicting values - perfect for demonstrating real-world data integration challenges.

```
In [1]: from pathlib import Path

# Setup paths
def get_repo_root():
    """Get repository root directory."""
    current = Path.cwd()
    while current != current.parent:
        if (current / 'pyproject.toml').exists():
            return current
        current = current.parent
    return Path.cwd()
```

Project Phase 2

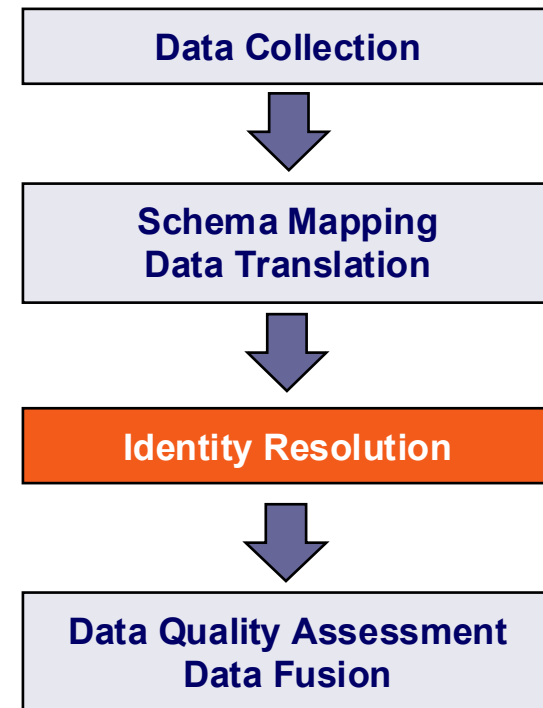
Project Phase 2: Identity Resolution

Duration: October 29th – November 12th

Tasks: Use PyDI to

1. Identify records in different data sets that describe the same real-world entity
2. Experiment with different combinations of similarity measures for matching rules and learning-based matchers.
3. Use blocking to speed up the comparisons
4. Evaluate the quality of your approach (F1 / Reduction Ratio)

Result: Correspondences between records in different data sets that describe the same entity



Prepare the Inputs: Check Your Data

- Your input is the output of your Schema Mapping phase
 - Schemata are aligned
 - Unique IDs are in place
- Are there duplicates in your data?
 - At least **1000 entities** should be contained in at least **two datasets**.
- Is there enough attribute overlap?
 - At least **5 attributes** should be contained in at least **two datasets**.
- Which combination of attributes can you use to detect duplicates?
 - name/title, creation/founding date, location/ address, height, colour, ...

Prepare the Inputs: Create Ground Truth

- You need a
 - Training set (learn matchers, select few-shot examples)
 - Validation set (evaluate choices, tune parameters)
 - Test set (final unbiased evaluation)
- Make labeled sets big enough
 - Each one at least 1% (or 300 pairs, if your datasets are huge) of entities
- You need labeled sets for all dataset pairs you want to match
 - At least 2 train, 2 validation, 2 test overall to create transitive closure
- Proceed iteratively
 - Create small labeled sets, then go through the whole exercise, then come back to improve the labeled sets by adding corner cases (and fixing errors)

The Ground Truth

- To train and evaluate identity resolution algorithms, you need manually verified **labeled sets of record pairs**
 - .csv files containing pairs of (comma-separated) IDs of entities that match **and** do not match
- The files includes non-trivial cases

```
gs_academy_awards_2_actors_test.csv:  
  
academy_awards_4529,actors_2,TRUE  
academy_awards_4500,actors_3,TRUE  
academy_awards_4421,actors_82,FALSE  
academy_awards_4475,actors_4,TRUE  
academy_awards_3305,actors_21,FALSE
```

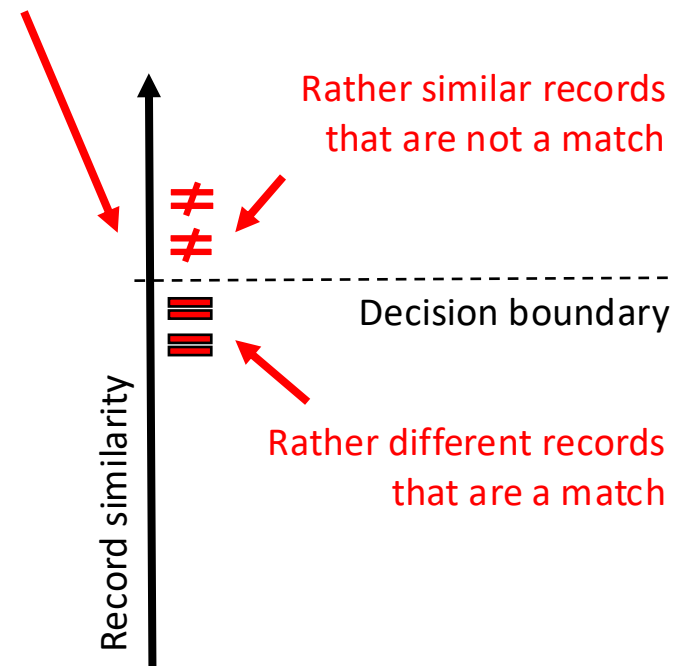
```
<movie>  
  <id>actors_141</id>  
  <title>Forest Gump</title>  
  <actors>  
    <actor>  
      <name>Tom Hanks</name>  
      <birthday>1956-01-01</birthday>  
      <birthplace>California</birthplace>  
    </actor>  
  </actors>  
  <date>1995-01-01</date>  
</movie>
```

```
<movie>  
  <id>academy_awards_902</id>  
  <title>Forrest Gump</title>  
  <director>  
    <name>Robert Zemeckis</name>  
  </director>  
  <actors>  
    <actor>  
      <name>Tom Hanks</name>  
    </actor>  
    <actor>  
      <name>Gary Sinise</name>  
    </actor>  
  </actors>  
  <date>1994-01-01</date>  
  <oscar>yes</oscar>  
</movie>
```

Diagram illustrating two XML snippets for the movie "Forrest Gump". The left snippet shows a record with ID "actors_141", title "Forest Gump", and a single actor "Tom Hanks" with birthdate "1956-01-01" and birthplace "California". The right snippet shows a record with ID "academy_awards_902", title "Forrest Gump", director "Robert Zemeckis", and two actors: "Tom Hanks" and "Gary Sinise". The date is "1994-01-01" and it has an "oscar" attribute set to "yes". Blue arrows point from the title and date fields of the left snippet to the corresponding fields in the right snippet, highlighting differences in the data.

The Ground Truth

- To create a labeled set, manually label a set of record pairs as **matches** or **non-matches** including **corner cases**
- Rule of thumb for creating a *suitable* labeled set with *acceptable* manual effort:
 1. match records using several simple matching techniques (goal: avoid selection bias) and sort record pairs according to their similarity
 2. if available, use information about likely matches (e.g. ISBN or GTIN numbers that exist in multiple sources)
 3. manually verify a fair amount of the resulting pairs (e.g. >300 pairs) including
 1. matching record pairs (randomly chosen, 20% of GS)
 2. corner case matches and non-matches (50% of GS)
 3. non-matching record pairs (randomly chosen, 30% of GS)



Identity Resolution in the Final Project Report

- Results of Phase 2 will be part of your final report
- Make sure you know/make notes on
 1. Content and size of your labeled sets
 - Which classes/data sets are included?
 - What „corner cases“ did you include?
 2. Which matching rules and matchers did you try?
 - What happens with P/R/F1?
 - Which attribute comparators / similarity measures did you use?
 3. What blockers have you tried?
 - What happened with runtime and number of matches?
 - What blockers / blocking functions have you used?
 - How do P/R/F1 change, and why?
- Note also that Phase 2 output is Exercise 3 (Data Fusion) input

Final Report: Tables describing your Experiments

1. Please add the following table to your final report and presentation slides:

#	Matcher	Blocker	P	R	F1	# Corr	Time
1	Rule1:Title&Year	No Blocking	0.71	0.95	0.82	10.230	90 min
2	Rule1:Title&Year	StandardYear	0.71	0.73	0.72	9.609	18 sec
3	Rule1:Title&Year	SNBYear	0.71	0.89	0.79	10.215	50 sec
4	Rule2:Title&Actors	SNBYear	0.81	0.89	0.83	9.919	19 sec

2. Please also report the group size distribution:

Group Size	Frequency
2	43
3	103
4	2

...and now

1. Prepare your labeled sets
2. Use PyDI and
 - Load your datasets
 - Define blockers
 - Define your matching rules and matchers
 - Run the evaluation
 - Learn matching rules (MLBasedMatcher)
 - (extra) Use LLM-based matching

