Exercise 3: Data Fusion

Web Data Integration

Agenda

1. Overview
2. Prepare Your Gold Standard
3. Fuse Your Data
    1. Apply conflict resolution functions
    2. Measure accuracy with respect to gold standard
4. Final Report and Final Exam

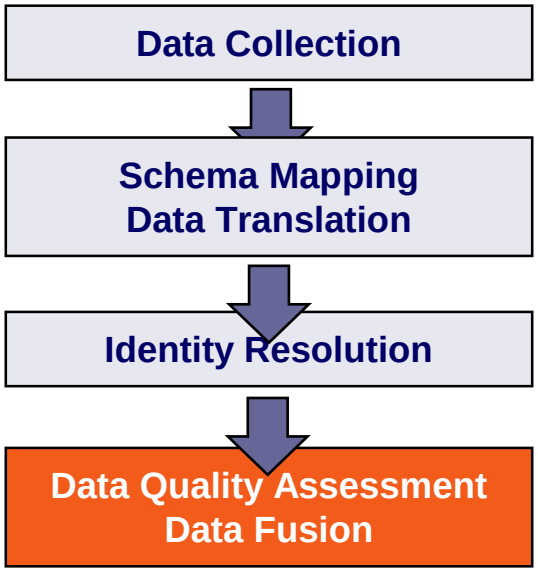
1. Overview

- Project Phase III: Data Fusion

  Duration: November 15th – November 30th

  Tasks: Extend Java project template to

  1. Merge data and resolve data conflicts
  2. Experiment with different conflict resolution functions
  3. Measure the density and accuracy of the final fused data set



  Results:

  1. Fused data set in which each real-world entity is described by only a single record and these records contain no data conflicts
  2. Project report (12 pages) summarizing the results of the phases 1-3

Select Data for Fusion Experiments

- Your input is the output of Exercises 1 and 2
  - schema is aligned, unique IDs are in place, identity resolution is done
- What data is suitable for fusion experiments?
  - Select at least 2 datasets for which
    1. attribute intersection is big enough
       - you should be able to fuse data for >= 5 attributes
    2. quality of identity resolution is good enough
    3. it makes sense to apply different fusion functions to the attributes
       - compare different functions
       - experiment with using provenance data vs. no provenance data

2. Prepare Your Gold Standard

- Your gold standard should contain
  - >= 15 entities
  - >= 4 attributes per entity
- Manually look up the correct values (in an external data source)!
- The Gold Standard uses your target schema
  - Use IDs from one of your data sets, so the evaluator can find the correct records!
- Don’t create ambiguities!
  - If records a, b, c are the same according to the identity resolution
  - Only one corresponding record can appear in the gold standard
  - Choose a, b or c as ID in the gold standard

3. Fuse Your data

- Use Java template project to run the data fusion experiments

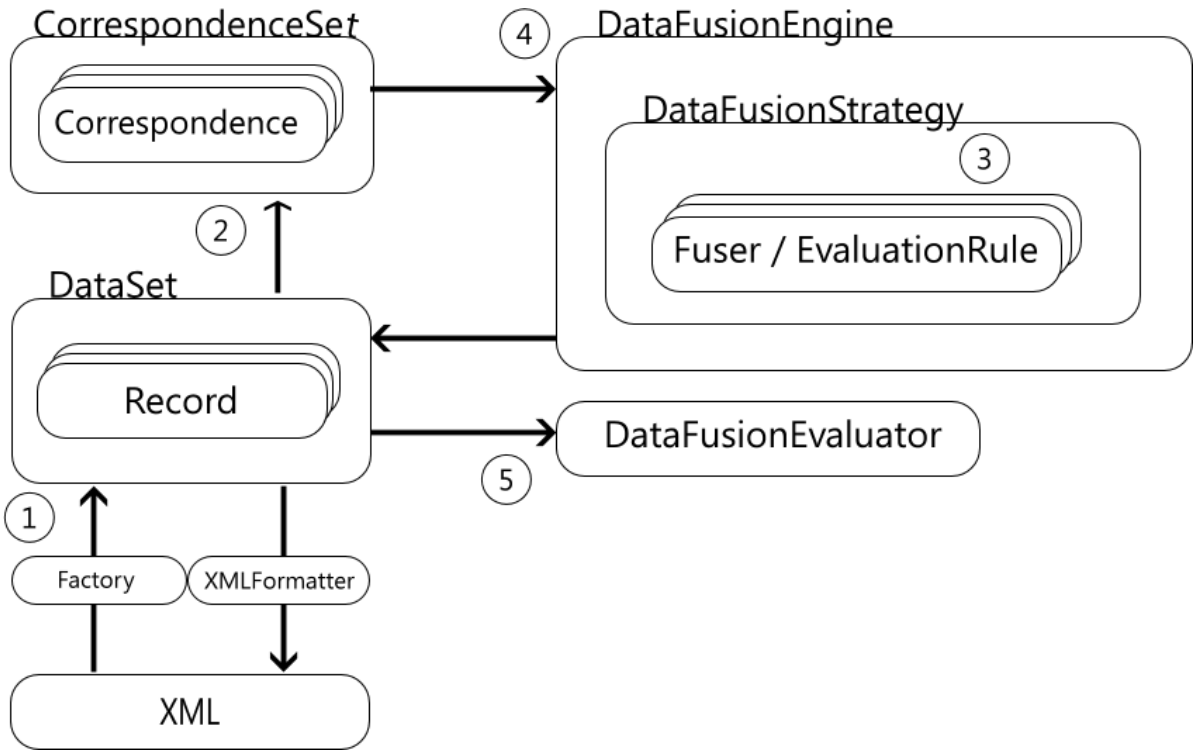
1. Try different conflict resolution functions for your attributes
2. Calculate the consistency of your input data
3. Compare density of your data sets before and after the fusion
4. Evaluate with your Gold Standard for Accuracy

The Java Template Project

- Download the .zip of the project from the course page
  - http://dws.informatik.uni-mannheim.de/en/teaching/courses-for-master-candidates/ie-670-web-data-integration/
- Unzip it and look at the sample input files in **\data**
  - .xml input datasets in **input** folder
  - .csv output of exercise 2 in **correspondences** folder
  - gold.xml in **goldstandard** folder
- We have implemented for you lots of things:
  - loading and analyzing your input datasets
  - computing the evaluation metrics density, consistency, and accuracy
  - various conflict resolution functions
  - comparing fusion results to a gold standard

Project Template Walkthrough: Movie Use Case

1. Data Sets
2. Correspondences
3. Fusion Strategy
4. Fusion Engine
5. Evaluation



Extend your Object Model

- Make your object model fusible, implement the *Fusible* interface
  - Provides meta data about the model for generating reports
- Or extend the *AbstractRecord* class, which implements both *Matchable* and *Fusible*

```
public class Movie
 extends AbstractRecord<Attribute> {

 public Movie(
  String identifier,
  String provenance) {
  super(identifier, provenance);
  actors = new LinkedList<>();
 }

 private String title;
 private String director;
 private LocalDateTime date;
 private List<Actor> actors;

 public String getTitle() {
  return title;
 }
 public void setTitle(String title) {
  this.title = title;
 }
...
}
```

Extend your Object Model

- Implement a *FusibleFactory*
  - It will create the fused objects

```
public class FusibleMovieFactory implements FusibleFactory<Movie, Attribute> {

 @Override
 public Movie createInstanceForFusion(RecordGroup<Movie, Attribute> cluster) {

  List<String> ids = new LinkedList<>();

  // collect the ids of all records that are fused in this group
  for (Movie m : cluster.getRecords()) {
   ids.add(m.getIdentifier());
  }

  // sort and merge the ids to create an id for the fused record
  Collections.sort(ids);

  String mergedId = StringUtils.join(ids, '+');

  // create the fused record
  return new Movie(mergedId, "fused");
 }

}
```

Loading Data into fusible data sets

- Load data using the FusibleDataSet class
  - Extends the DataSet class

```
// Load the Data into FusibleDataSet
FusibleDataSet<Movie, Attribute> ds1 = new FusibleHashedDataSet<>();
new MovieXMLReader().loadFromXML(new File("data/input/academy_awards.xml"), "/movies/movie", ds1);
```

  - Allows you to add data set meta data

```
ds1.setScore(3.0);
ds1.setDate(LocalDateTime.parse("2012-01-01", formatter));
```

  - Calculates density report

```
ds1.printDataSetDensityReport();
```

```
DataSet density:                0,58
Attributes densities:
        Actors:                 0,23
        Date:                   1,00
        Title:                  1,00
        Director:               0,09
```

Loading Correspondences

- Use the *CorrespondenceSet* class to load all your correspondences
  - Call *loadCorrespondences* multiple times!

```
// load the correspondences
CorrespondenceSet<FusableMovie> correspondences = new CorrespondenceSet<>();

correspondences.loadCorrespondences( new File("correspondences.csv"),
                                     ds1,
                                     ds2);
```

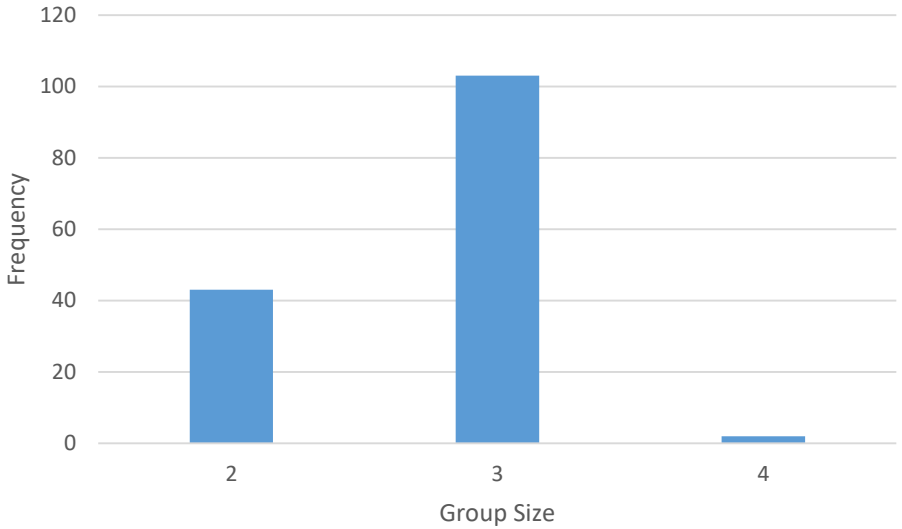
  - The order of the data sets (2nd and 3rd parameter) must match the order of Ids in the correpondence file!

Correspondence Group Distribution

- Create correspondence group distribution
  - All correspondences from the identity resolution are combined
  - All records that are believed to describe the same real-world entity end up in a group (transitive!)

```
// write group size distribution
correspondences.printGroupSizeDistribution();
```

```
Group Size Distribution of 148 groups:
Group Size | Frequency
_____________________________
2          |          43
3          |          103
4          |          2
```



Defining your Fusion Strategy

- Use the *DataFusionStrategy* class to define how each attribute is fused
- For each attribute, you have to add a *Fuser* and an *EvaluationRule*
  - Fusers use a conflict resolution function to fuse the values for an attribute
  - EvaluationRules determine whether two values are equal

```
// define the fusion strategy
DataFusionStrategy<Movie,Attribute> strategy =
 new DataFusionStrategy<>(new FusibleMovieFactory());

// add attribute fusers
// Note: The attribute name is only used for printing the reports
strategy.addAttributeFuser(  "Title",
                             new TitleFuser(),
                             new TitleEvaluationRule());
```

Defining a Fuser

- A fuser defines a conflict resolution function
  - That decides which of many values to choose
- And three additional functions
  - If a record has a value that can be fused
  - How to get the value for a record
  - How to assign the fused value to the fused record

```
public class TitleFuser extends AttributeValueFuser<String, Movie, Attribute> {
 public TitleFuser() {
  super(new LongestString<Movie>());
 }

 @Override
 public boolean hasValue(Movie record) {
  return record.hasValue(Movie.TITLE);
 }

 @Override
 public String getValue(Movie record) {
  return record.getTitle();
 }
```

Define the conflict resolution function

Does *record* have a value for this fuser?

Get *record*'s value for this fuser?

Defining a Fuser

- A fuser defines a conflict resolution function
  - That decides which of many values to choose
- And three additional functions
  - If a record has a value that can be fused
  - How to get the value for a record
  - How to assign the fused value to the fused record

```
@Override
public void fuse(RecordCluster<Movie> group,
 Movie fusedRecord) {

 FusedValue<String, Movie> fused = getFusedValue(group);

 fusedRecord.setTitle(fused.getValue());

 fusedRecord.setAttributeProvenance(Movie.TITLE, fused.getOriginalIds());
}
```

Get the fused value

Assign the fused value to the fused record

Add provenance info to the fused record

Implemented Conflict Resolution Functions

Numeric Functions	
Average	Calculates the average of all values
Median	Calculates the median of all values
String Functions	
Longest String	Chooses the longest string
Shortest String	Chooses the shortest string
List Functions	
Union	Creates the union of all values (lists)
Intersection	Creates the intersection of all values (lists)
Functions that use Provenance Metadata	
FavourSources	Chooses the value from the data source with the highest score
MostRecent	Chooses the most up-to-date value
Data Type Independent Functions	
Voting	Chooses the most frequent value
ClusteredVote	Chooses the centroid of the largest value cluster

Specifying your Evaluation Rules

- Extend the *EvaluationRule* class
  - defines which values should be seen as equal
  - rules are applied for the calculation of consistency and accuracy
- It might be OK to tolerate
  - +/- 2% for numeric data like temperature, population
  - edit distance 1 for people names but likely not for movie names
- Ideal case: Complete equality, no tolerance needed

```
public class TitleEvaluationRule extends EvaluationRule<Movie, Attribute> {

SimilarityMeasure<String> sim = new TokenizingJaccardSimilarity();

@Override
public boolean isEqual(Movie record1, Movie record2) {
  // the title is correct if all tokens are there, but the order does not matter
  return sim.calculate(record1.getTitle(), record2.getTitle()) == 1.0;
 }
}
```

Running the Fusion

- Use the DataFusionEngine class
  - calculates value consistency

```
// create the fusion engine
DataFusionEngine<Movie,Attribute> engine = new DataFusionEngine<>(strategy);

// calculate attribute consistency
engine.printClusterConsistencyReport(correspondences, null);
```

```
Attribute Consistencies:
Actors: 0,43
Date: 0,00
Title: 0,94
Director: 1,00
```

  - And fuses your data

```
// run the fusion
FusableDataSet<Movie,Attribute> fusedDataSet = engine.run(correspondences, null);
```

Writing the fused dataset

- Extend XMLFormatter to write your dataset to an XML file

```
public class MovieXMLFormatter extends XMLFormatter<Movie> {
 @Override
 public Element createRootElement(Document doc) {
  return doc.createElement("movies");
 }

 @Override
 public Element createElementFromRecord(FusableMovie record, Document doc) {
  Element movie = doc.createElement("movie");

  movie.appendChild(createTextElement("id",
   record.getIdentifier(), doc));

  movie.appendChild(createTextElementWithProvenance("title",
   record.getTitle(),
   record.getMergedAttributeProvenance(Movie.TITLE), doc));
…
  return movie;
 }
}
```

The fused data set

```
<?xml version="1.0" encoding="UTF-8"?>
<movies>
  <movie>
    <id>academy_awards_2334+actors_39+golden_globes_2000</id>
    <title provenance="golden_globes_2000">Who's Afraid Of Virginia Woolf</title>
    <director provenance="academy_awards_2334">Mike Nichols</director>
    <date provenance="golden_globes_2000+actors_39">1967-01-01T00:00</date>
    <actors provenance="academy_awards_2334+golden_globes_2000+actors_39">
      <actor>
        <name>George Segal</name>
      </actor>
      <actor>
        <name>Richard Burton</name>
      </actor>
      <actor>
        <name>Elizabeth Taylor</name>
      </actor>
      <actor>
        <name>Sandy Dennis</name>
      </actor>
    </actors>
  </movie>
```

Evaluating Your Fusion Result

- Use the DataFusionEvaluator class
  - needs your fusion strategy!
  - accepts a DataSet as gold standard
  - returns the accuracy of your result

```
// load the gold standard
DataSet<Movie, Attribute> gs = new FusibleHashedDataSet<>();

new MovieXMLReader().loadFromXML(       new File("data/goldstandard/fused.xml"),
                                        "/movies/movie", gs);

// evaluate
DataFusionEvaluator<Movie, Attribute> evaluator = new DataFusionEvaluator<>(strategy );

evaluator.setVerbose(true);

double accuracy = evaluator.evaluate(fusedDataSet, gs, null);
```

4. Requirements for the Final Project Report

- 12 pages (sharp!) – counted without title page, table of content, literature list
  - Every extra page (including appendix pages) will reduce your mark by 0.33
- Due to Friday, 30th November 2018, 23:59
  - Send by **email to Chris, Oliver and Anna**
- You must use the **DWS master thesis layout**
  - http://dws.informatik.uni-mannheim.de/en/thesis/masterthesis/
- Also **submit**
  1. your code and
  2. (a subset) of your data
- Please cite sources properly if you use any
  - Preferred citation style [Author, year]

Final Report : Content

- Your final report should contain
  1. Results of Phase 1: Data Translation
  2. Results of Phase 2: Identity Resolution
  3. Results of Phase 3: Data Fusion
  4. Summary of the overall result

Data Translation in the Final Report

- Your report should contain
  1. Profiling results describing your input data sets
     - e.g. updated versions of the tables that you created for your project proposal

Dataset	Source (*)	format	class (**)	# of entities	# of attributes	list of attributes (***)
IMDB	Download URL	csv	movie	17,000	10	title, director, year, ...
DBpedia	dbpedia.org/sparql	xml	actor	23,500	8	name, birthDate (MV), activeYears,...
Freebase	Download URL	csv	actor	11,000	14	given_name, surname, spouse (MV)
.....						

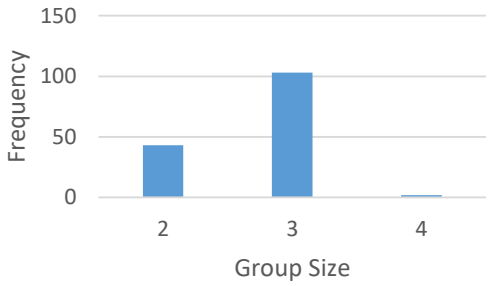
Class name	Attribute name	Datasets in which attribute is found
movie	name	dataset1, dataset2, dataset3, dataset4
movie	director	dataset1, dataset3
movie	year	dataset2, dataset3, dataset4
.....		

  2. Your consolidated schema and how you created it
  3. Which transformations you used and why
     - if there was any information you could not transform

Identity Resolution in the Final Report

- Your report should contain
  1. Content and size of your gold standard and procedure used to create it
  2. Which matching rules did you try?
     - Discuss what happened with P/R and F1?
     - Please include a table comparing the results of the different matching rules
  3. Which blocking methods did you try?
     - Report and discuss the change in runtime, number of matches, and reduction ratio. How do P/R/F1 change?
     - Please include a table comparing the results of the different blocking methods that you tested
  4. What's the group size distribution of your result?
  5. An analysis of the errors that remain when applying your best matching rule.

#	Matching Rule	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



Data Fusion in the Final Report

- Your report should contain
  1. Which datasets your selected for fusion?
  2. What kind of provenance data you added?
  3. What was the density of your input and the merged datasets?
  4. How consistent were your datasets?
  5. Size and content of your gold standard and how you created it
  6. Which conflict resolution functions you tried for each attribute
     - Whether your define your own conflict resolution functions
  7. Which accuracy did the different conflict resolution functions deliver? What was the best function for each attribute?
     - Please include a table comparing the accuracies reached by the different resolution functions.
  8. An analysis of the errors that remain after applying your best resolution function.

Summary of Overall Results in Final Report

Please conclude your report with a summary answering the following three questions:

1. How many additional entities did you add compared to the largest of your input datasets?
2. How much did you increase the density of your data compared to the largest of your input datasets?
3. What is the overall accuracy of your final dataset according to your gold standard?

Final Report: Important

- Balance your content between the 3 exercises
  - not 10 pages on identity resolution and 2 pages on the rest
- If you have done something cool – write about it!
  - it is highly unlikely we dig it out of your code ourselves
- We are strict about the 12 pages limit. Thus,
  - include lots of tables to show us what you have tried
  - briefly discuss the results of each of your experiments
  - do not repeat theoretical stuff from the slides (e.g. definition of X)
  - we will reduce your mark by 0.33 for each extra page
    no matter how interesting it is!

Task

1. Open and run the provided Java project
2. Inspect the profiling results
    1. What is the potential of every dataset to fill missing values within the other datasets?
    2. How many real world elements are presented by more than one element?
    3. How is it possible to have 4 conflicting elements when we have only 3 datasets?
    4. Which attributes do you expect to be more conflicting?
3. Create your own value-based fusion strategy that selects the longest title value
4. Which source is the most trustworthy as far as the date attribute is concerned?

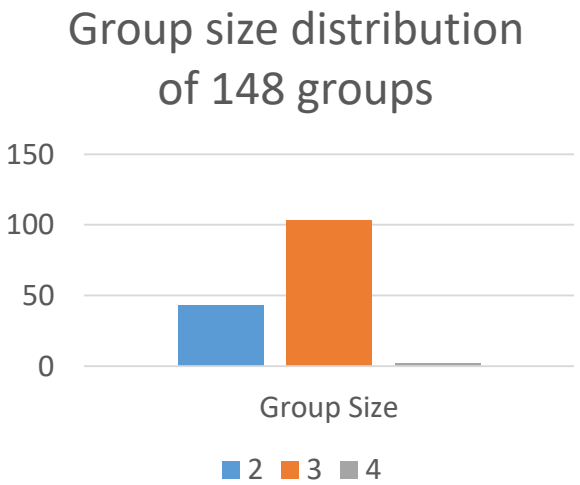
Solutions

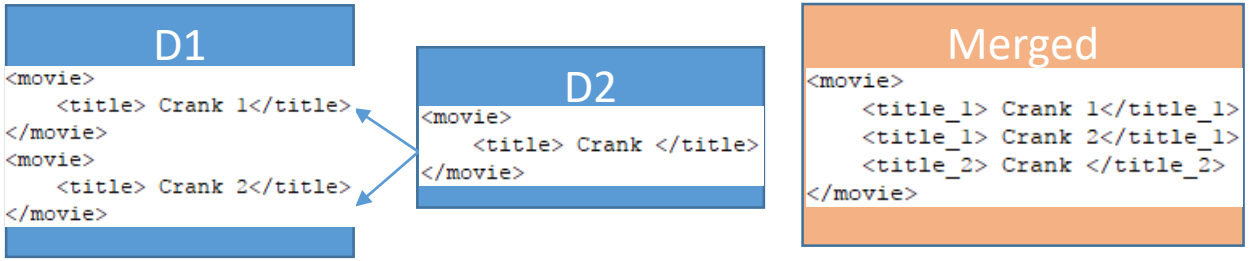
Interpreting profiling results

Dataset	Number of Elements	Overall Density	Attribute Density			
			Actors	Date	Title	Director
academy_awards.xml	4580	0.58	0.23	1.00	1.00	0.09
actors.xml	151	0.75	1.00	1.00	1.00	0.00
golden_globes.xml	2279	0.78	0.98	1.00	1.00	0.14

Good for missing values

High overlap: More potential for conflicts





Solutions

Create your own value-based fusion strategy that selects the longest title value

```
public class TitleFuserLongestString extends AttributeValueFuser<String, Movie,
Attribute> {

    public TitleFuserLongestString() {
        super(new LongestString<Movie, Attribute>());
    }

    public void fuse(RecordGroup<Movie, Attribute> group, Movie fusedRecord,
    Processable<Correspondence<Attribute, Matchable>> schemaCorrespondences,
    Attribute schemaElement) {
        ...
    }
    public boolean hasValue(Movie record, Correspondence<Attribute, Matchable>
    correspondence) {
        ...
    }
    protected String getValue(Movie record, Correspondence<Attribute, Matchable>
    correspondence) {
        ...
    }
```

Solutions

Which source is the most trustworthy as far as the date attribute is concerned?

- Adjust the dataset scores
- Use a conflict resolution function that uses these scores

```
strategy.addAttributeFuser(
        Movie.DATE,
        new DateFuserFavourSource(),
        new DateEvaluationRule());
```

- Check how the accuracy changes

```
ds1.setScore(1.0);
ds2.setScore(2.0);
ds3.setScore(3.0);
```

Attribute-specific accuracy: 0.00

```
ds1.setScore(3.0);
ds2.setScore(1.0);
ds3.setScore(2.0);
```

Attribute-specific accuracy: 0.95

Final Exam

- Date and Time of WDI Exam
  - **Wednesday, 17.12.2018, 08:30, Room A5 B1.44**
- Format
  - 5-6 open questions that show that you have understood the theory part of the lecture
    - all lecture slide sets including structured data on the Web and data exchange formats + query languages

      XPath and SPARQL
  - Duration: 60 minutes

...and now

- Get the template project and
  - Define your inputs
  - Experiment with creating the merged dataset, and density and consistency evaluation metrics
  - Define your conflict resolution functions
  - Define your gold standard
  - Experiment with data fusion and accuracy evaluation metrics
- **Write your final report**
- **Repeat the theory parts in order to be ready for the final exam**