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 8\textsuperscript{st} – December 1\textsuperscript{th}  
  
  **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
       – at least one attribute should not be of type string
    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
  >= 5 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

• 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*

```java
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

```java
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

```java
// 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

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

• Calculates density report

```
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!

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

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

• The order of the data sets (2\textsuperscript{nd} and 3\textsuperscript{rd} parameter) must match the order of Ids in the correspondence 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();
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

```java
// 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

```java
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
    protected String getValue(Movie record) {
        return record.getTitle();
    }
}
```
Defining a Fuser

- A fuser defines a conflict resolution
  - 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

```java
@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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>Calculates the average of all values</td>
</tr>
<tr>
<td>Median</td>
<td>Calculates the median of all values</td>
</tr>
</tbody>
</table>

### String Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longest String</td>
<td>Chooses the longest string</td>
</tr>
<tr>
<td>Shortest String</td>
<td>Chooses the shortest string</td>
</tr>
</tbody>
</table>

### List Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Union</td>
<td>Creates the union of all values (lists)</td>
</tr>
<tr>
<td>Intersection</td>
<td>Creates the intersection of all values (lists)</td>
</tr>
</tbody>
</table>

### Functions that use Provenance Metadata

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FavourSources</td>
<td>Chooses the value from the data source with the highest score</td>
</tr>
<tr>
<td>MostRecent</td>
<td>Chooses the most up-to-date value</td>
</tr>
</tbody>
</table>

### Data Type Independent Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voting</td>
<td>Chooses the most frequent value</td>
</tr>
<tr>
<td>ClusteredVote</td>
<td>Chooses the centroid of the largest value cluster</td>
</tr>
</tbody>
</table>
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

```java
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

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

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

- And fuses your data

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

Attribute Consistencies:
- Actors: 0.43
- Date: 0.00
- Title: 0.94
- Director: 1.00
Writing the fused dataset

- Extend XMLFormatter to write your dataset to an XML file

```java
public class MovieXMLFormatter extends XMLFormatter<

@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;
}
```
<?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>
</movies>
Evaluating Your Fusion Result

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

```java
// 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,
                new RecordGroupFactory<Movie, Attribute>());

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, 1st December 2017, 23:59**
  • Send by email to Chris, Oliver and Anna

• You must use the **DWS master thesis layout**

• 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]
• Your final report should contain
  1. Results of Exercise 1: Data Translation
  2. Results of Exercise 2: Identity Resolution
  3. Results of Exercise 3: Data Fusion
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
  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. Whether you have learned matching rule using Rapidminer and whether this rule was better than your handwritten one
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 you 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.
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?
### Interpreting Profiling Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of Elements</th>
<th>Overall Density</th>
<th>Attribute Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>academy_awards.xml</td>
<td>4580</td>
<td>0.58</td>
<td>Actors: 0.23</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date: 1.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Title: 1.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Director: 0.09</td>
</tr>
<tr>
<td>actors.xml</td>
<td>151</td>
<td>0.75</td>
<td>1.00</td>
</tr>
<tr>
<td>golden_globes.xml</td>
<td>2279</td>
<td>0.78</td>
<td>0.98</td>
</tr>
</tbody>
</table>

**Group size distribution of 141 groups**

- Good for missing values
- High overlap

**Example XML**

- **D1**:
  ```xml
  <movie>
  <title>Crank 1</title>
  </movie>
  <movie>
  <title>Crank 2</title>
  </movie>
  ``

- **D2**:
  ```xml
  <movie>
  <title>Crank 1</title>
  </movie>
  <movie>
  <title>Crank 2</title>
  </movie>
  ```

- **Merged**:
  ```xml
  <movie>
  <title>Crank 1</title>
  <title>Crank 2</title>
  </movie>
  ```
Create your own value based fusion strategy that selects the longest title value

```java
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) {
        ...
    }
}
```
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

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

- Check how the accuracy changes

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

Attribute-specific accuracy: 0.38

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

Attribute-specific accuracy: 0.63
Final Exam

• Date and Time of WDI Exam
  • Wednesday, 13.12.2017, 08:30, A5 B243

• 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

IT'S... TIME!
To Go to Work!