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 18\textsuperscript{st} – December 6\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 3 datasets for which
    1. attribute intersection is big enough
      – you should be able to fuse data for \( \geq 4 \) attributes per class
      – 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
  - \( \geq 5 \) entities
  - \( \geq 4 \) 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 `usecase/movie`
  • .xml input datasets in `datasets` folder
  • .csv output of exercise 2 in `duplicates` 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
Project Template: Setup

- The DataFusion project references the IdentityResolution project
  - Make sure it is referencing your code and not the empty template!
- You will re-use and extend the object model for representing your use case data

- Alternative: Integrate (!) both projects into one
  - Add all classes from the DataFusion template to your project
  - Merge the pom.xml files
  - Then you can modify your classes instead of extending them
Project Template: Extend your Object Model

- Extend your object model, implement the *Fusible* interface
  - Provides meta data about the model for generating reports
- Extend your factory, implement the *FusableFactory* interface
  - Define how to create a fused instance

```java
public class FusableMovieFactory
    extends MatchableFactory<FusableMovie>
    implements FusableFactory<FusableMovie> {
    ...
    
    @Override
    public FusableMovie createInstanceForFusion(
        RecordCluster<FusableMovie> cluster) {
        ...
        return new FusableMovie(mergedId, "fused");
    }
}
```

```java
public interface Fusable {
    Collection<String>.getAttributeNames();
    boolean hasValue(String attributeName);
}
```

```java
public interface FusableFactory
    <RecordType extends Matchable & Fusable> {
    
    abstract RecordType
    createInstanceForFusion(
        RecordCluster<RecordType> cluster);
}
```
Project Template: Loading Data

- Load data using the FusableDataSet class
  - Extends the DataSet class
  - Adds provenance metadata
  - Calculates density report

```java
public class FusableDataSet
        <RecordType extends Matchable & Fusable>
    extends DataSet<RecordType> {
        ...

        public void setScore(double score) { ... }
        public void setDate(DateTime date) { ... }

        public double getDensity() { ... }
        public Map<String, Double> getAttributeDensities() { ... }
        public void printDataSetDensityReport() { ... }

        public void writeXML(
            File outputFile,
            XMLFormatter<RecordType> formatter) { ... }
    }
```

FusableDataSet<FusableMovie> ds1 = new FusableDataSet<>();

```
ds1.loadFromXML(
    new File("academy_awards.xml"),
    new FusableMovieFactory(),
    "/movies/movie"
);

// set dataset metadata
ds1.setScore(1.0);
ds1.setDate(
    DateTime.parse("2012-01-01")
);
```

```
DataSet density: 0,58
Attributes densities:
    Actors: 0,23
    Date: 1,00
    Title: 1,00
    Director: 0,09
```
Adding Provenance Information to Your Data Sets

- Provenance information can be important to get good fusion results
  - think about which kind of provenance information exists in your use case
  - and how you could use this information for conflict resolution

- Already implemented dataset-level provenance:
  - Date: From which point in time is the data?
    - Population numbers from around 500 B.C. will not lead to correct values for today
  - Score: Rank your datasets
    - Can be calculated in many different ways, think about what makes sense for your use case

- Feel free to extend the provenance part
  - There are many possibilities, i.e. include the rating/trust of the author who create a record in your data set

FusableDataSet<FusableMovie> ds1 = new FusableDataSet<>();

// set dataset metadata
ds1.setScore(1.0);
ds1.setDate(
DateTime.parse("2012-01-01"));
Project Template: Load Correspondences

- Use the CorrespondenceSet class to load all your correspondences
  - Call loadCorrespondences multiple times!
  - Dataset order (parameters) must match the order in the correspondence file!

```java
public class CorrespondenceSet<RecordType extends Matchable & Fusable> {
    ...

    public void loadCorrespondences(
        File correspondenceFile,
        FusableDataSet<RecordType> first,
        FusableDataSet<RecordType> second) { ... }

    public Collection<RecordCluster<RecordType>> getRecordGroups() { ... }

    public void writeGroupSizeDistribution(
        File outputFile) { ... }
}
```

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

correspondences.loadCorrespondences(
    new File("correspondences.csv"),
    ds1,
    ds2);
```
Project Template: 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!)

```java
// write group size distribution
correspondences.writeGroupSizeDistribution(
    new File("group_size_distribution.csv"));
```

![Bar chart showing group size distribution](image)
Project Template: Define 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
public class DataFusionStrategy
    <RecordType extends Matchable & Fusable> {
    ...

    public void addAttributeFuser(
        String attributeName,
        AttributeFuser<RecordType> fuser,
        EvaluationRule<RecordType> rule) { ... }

    public RecordType apply(
        RecordGroup<RecordType> group) { ... }
}
```

```java
// define the fusion strategy
DataFusionStrategy<FusableMovie> strategy =
    new DataFusionStrategy<>(
        new FusableMovieFactory());

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

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

    @Override
    public void fuse(RecordCluster<FusableMovie> group,
                     FusableMovie fusedRecord) {
        FusedValue<String, FusableMovie> fused = getFusedValue(group);
        fusedRecord.setTitle(fused.getValue());
        fusedRecord.setAttributeProvenance(FusableMovie.TITLE, fused.getOriginalIds());
    }

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

    @Override
    protected String getValue(FusibleMovie record) {
        return record.getTitle();
    }
}
```

- Define the conflict resolution function
- Assign the fused value to the fused record
- Add provenance info to the fused record
- Interprets your Data Model for the Fusion Engine
## Already Implemented Conflict Resolution Functions

<table>
<thead>
<tr>
<th>Numeric Functions</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>String Functions</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>List Functions</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 Provanance Metadata

<table>
<thead>
<tr>
<th>Functions that use Provanance Metadata</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>Data Type Independent Functions</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>
Project Template: Specify 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<FusableMovie> {

    SimilarityMeasure<String> sim = new TokenizingJaccardSimilarity();

    @Override
    public boolean isEqual(FusableMovie record1, FusableMovie 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;
    }
}
```
Project Template: Create the Fusion Engine

- Use the DataFusionEngine class
  - can fuse your data
  - calculates value consistency

```java
public class DataFusionEngine
    <RecordType extends Matchable & Fusable> {

    public DataFusionEngine(
        DataFusionStrategy<RecordType> strategy) { ... }

    public FusableDataSet<RecordType>
        run(CorrespondenceSet<RecordType> correspondences) { ... }

    public Map<String, Double>
        getAttributeConsistencies(
            CorrespondenceSet<RecordType> correspondences) { ... }

    public void printGroupConsistencyReport(
        CorrespondenceSet<RecordType> correspondences) { ... }
}
```

// create the fusion engine
DataFusionEngine<FusableMovie>
    engine = new DataFusionEngine<> (strategy);

// calculate group consistency
engine.printGroupConsistencyReport (correspondences);

// run the fusion
FusibleDataSet<FusableMovie>
    fusedDataSet = engine.run (correspondences);

Attribute Consistencies:
Actors: 0,43
Date: 0,00
Title: 0,94
Director: 1,00
Project Template: Write your fused dataset

- Extend XMLFormatter to write your dataset to an XML file

```java
public abstract class XMLFormatter<RecordType> {

    public abstract Element createRootElement(Document doc);
    public abstract Element createElementFromRecord(RecordType record, Document doc);

    protected Element createTextElement(String name, String value, Document doc) {
        // implementation...
    }

    public class MovieXMLFormatter extends XMLFormatter<FusableMovie> {
        @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(createTextElement("title", record.getTitle(), doc));

            // other elements...

            return movie;
        }
    }
```
Project Template: Evaluate Your Fusion Result

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

```java
class DataFusionEvaluator<RecordType>
    extends Matchable & Fusable {

  DataFusionEvaluator(DataFusionStrategy<RecordType> strategy) {
    ...
  }

  double evaluate(DataSet<RecordType> dataset,
                  DataSet<RecordType> goldStandard) {
    ...
  }

  public class DataFusionEvaluator
    <RecordType extends Matchable & Fusable> {

    public DataFusionEvaluator(
        DataFusionStrategy<RecordType> strategy) {
      ...
    }

    public double evaluate(
        DataSet<RecordType> dataset,
        DataSet<RecordType> goldStandard) {
      ...
    }

    // load the gold standard
    DataSet<FusableMovie> gs = new FusableDataSet<>();
    gs.loadFromXML(
        new File("fused.xml"),
        new FusableMovieFactory(),
        "/movies/movie");

    // evaluate
    DataFusionEvaluator<FusableMovie>
        evaluator = new DataFusionEvaluator<> (strategy);

    double accuracy = evaluator.evaluate(
        fusedDataSet,
        gs);
  }
```
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 6 December 2015, 23:59**
  - Send by email to Chris and Oliver
- **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]
Final Report : Content

- 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 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.
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!
Final Exam

• Date and Time of WDI Exam
  • 17 December 2015, 17:00, 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
  • 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