ERBlox: Combining Matching Dependencies with Machine Learning for Entity Resolution

Leopoldo Bertossi

Carleton University
School of Computer Science
Institute for Data Science
Ottawa, Canada

bertossi@scs.carleton.ca

Prelude on Data Quality

Recent Approaches to Data Quality

- Data quality has many dimensions: consistency, completeness, accuracy, redundancy, freshness, ...

  All of them create in the end a problem of uncertainty in data

- Consistency has to do with satisfying semantic constraints, usually in the form of integrity constraints (ICs)

  ICs have been around for a long time ...

  They are used to capture the application semantics in the data model and database

  They have been studied in general and have wide application in data management

  Much fundamental/technical research has been developed

Methodologies for dealing with ICs are quite general and have broad applicability

- However, in many situations databases may be inconsistent wrt. a given set of ICs

  Getting rid of violations is sometimes possible, but sometimes impossible or too complex or undesirable

- Why not accepting inconsistency, live with it, and make the best we can out of our DB?

- Database repairing and consistent query answering (CQA) are newer contributions in this direction (more coming)

  And more generally, a contribution to a newer approach to data quality problems

- Data quality assessment (DQ) and data cleaning (DC) have been mostly: Ad-hoc, rigid, vertical, and application-dependent activities
- There is a lack of fundamental research in data quality assessment and cleaning
- Things are starting to change ...
- Recently, different forms of data quality constraints have been proposed and investigated
- They provide generic languages for expressing quality concerns

  Suitable for specifying adaptive and generic data quality assessment and data cleaning techniques

Characterizing Consistent Data wrt ICs

- What are the consistent data in an inconsistent database?

  What are the consistent answers to a query posed to an inconsistent database?

- (Arenas,Bertossi,Chomicki; PODS99) provided a precise definition

  Intuitively, the consistent data in an inconsistent database D are invariant under all minimal ways of restoring D's consistency

  Consistent data persists across all the minimally repaired versions of the original instance: the repairs of D

Example: For the instance D that violates

FD: $Name \rightarrow Salary$

$Employee$	$Name$	$Salary$
	$page$	5K
	$page$	8K
	$smith$	3K
	$stowe$	7K

Two possible (minimal) repairs if only deletions/insertions of whole tuples are allowed: D_1, resp. D_2

$Employee$	$Name$	$Salary$
	$page$	5K
	$smith$	3K
	$stowe$	7K

$Employee$	$Name$	$Salary$
	$page$	8K
	$smith$	3K
	$stowe$	7K

$(stowe, 7K)$ persists in all repairs: it is consistent information

$(page, 8K)$ does not (it participates in the violation of FD)

- A consistent answer to a query $\mathcal{Q}$ from a database D is one that can be obtained as a usual answer to $\mathcal{Q}$ from every possible repair of D wrt IC
  - $\mathcal{Q}_1 : Employee(x, y)?$

    Consistent answers: $(smith, 3K), (stowe, 7K)$
  - $\mathcal{Q}_2 : \exists y Employee(x, y)?$

    Consistent answers: $(page), (smith), (stowe)$

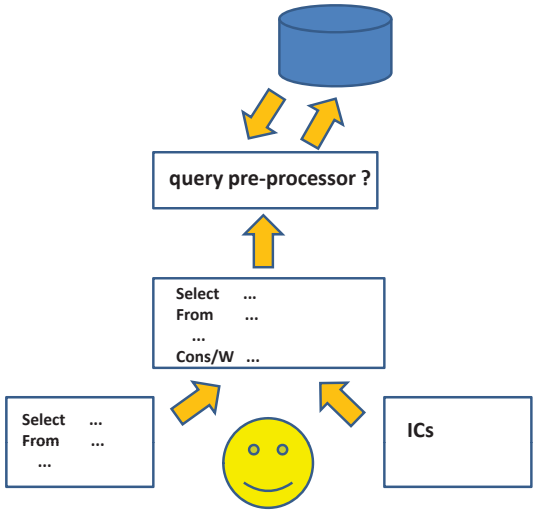
CQA may be different from classical data cleaning!

However, CQA is relevant for data quality; an increasing need in business intelligence

It also provides concepts and techniques for data cleaning

- Next DBMSs should provide more flexible, powerful, and user friendlier mechanisms for dealing with semantic constraints

  In particular, they should accept and answer queries requesting for consistent data



Why not an enhanced SQL?

SELECT Name, Salary
FROM Employee
CONS/W FD: Name –> Salary;

(FD not maintained by the DBMS)

- Paradigm shift: ICs are constraints on query answers, not on database states!

  A form of data cleaning wrt IC violation at query-answering time!

- Idea: For CQA avoid or minimize computation/materialization of repairs

- For some tractable cases of CQA, query rewriting algorithms have been developed

  New query can be posed/answered as usual to/from the original DB

$$\mathcal{Q}(x, y): \ Employee(x, y) \quad \mapsto$$
$$\mathcal{Q}'(x, y): \ Employee(x, y) \wedge \neg\exists z(Employee(x, z) \wedge z \neq y)$$

```
SELECT    Name, Salary    ↦    SELECT    Name, Salary
FROM      Employee;              FROM      Employee
                                 WHERE     NOT EXISTS (
                                   SELECT *
                                   FROM   Employee E
                                   WHERE  E.Name = Name AND
                                          E.Salary <> Salary);
```

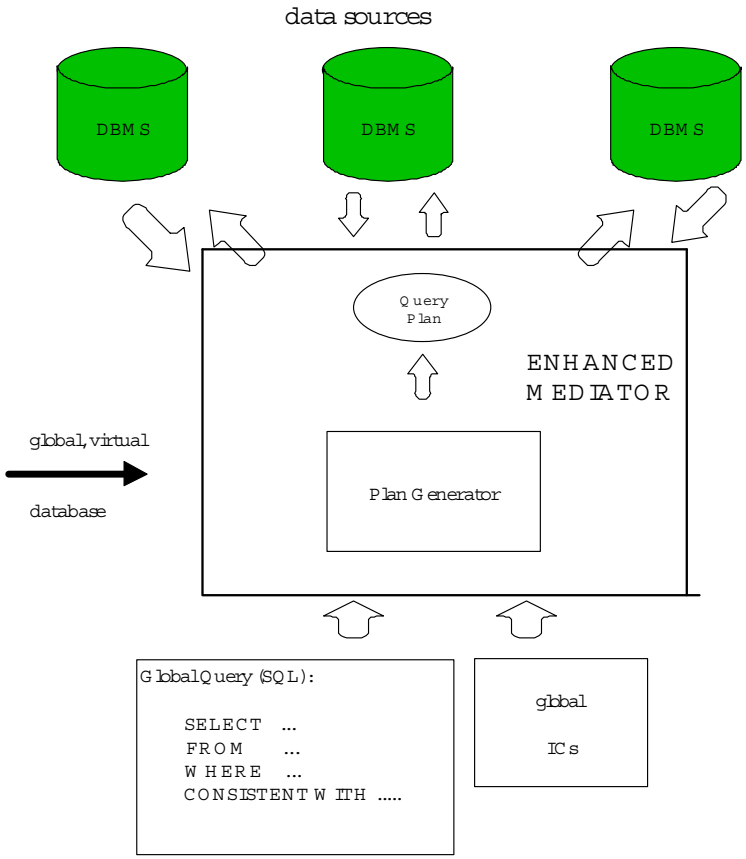
(retrieves employees with their salaries for which there is no other employee with the same name, but different salary)

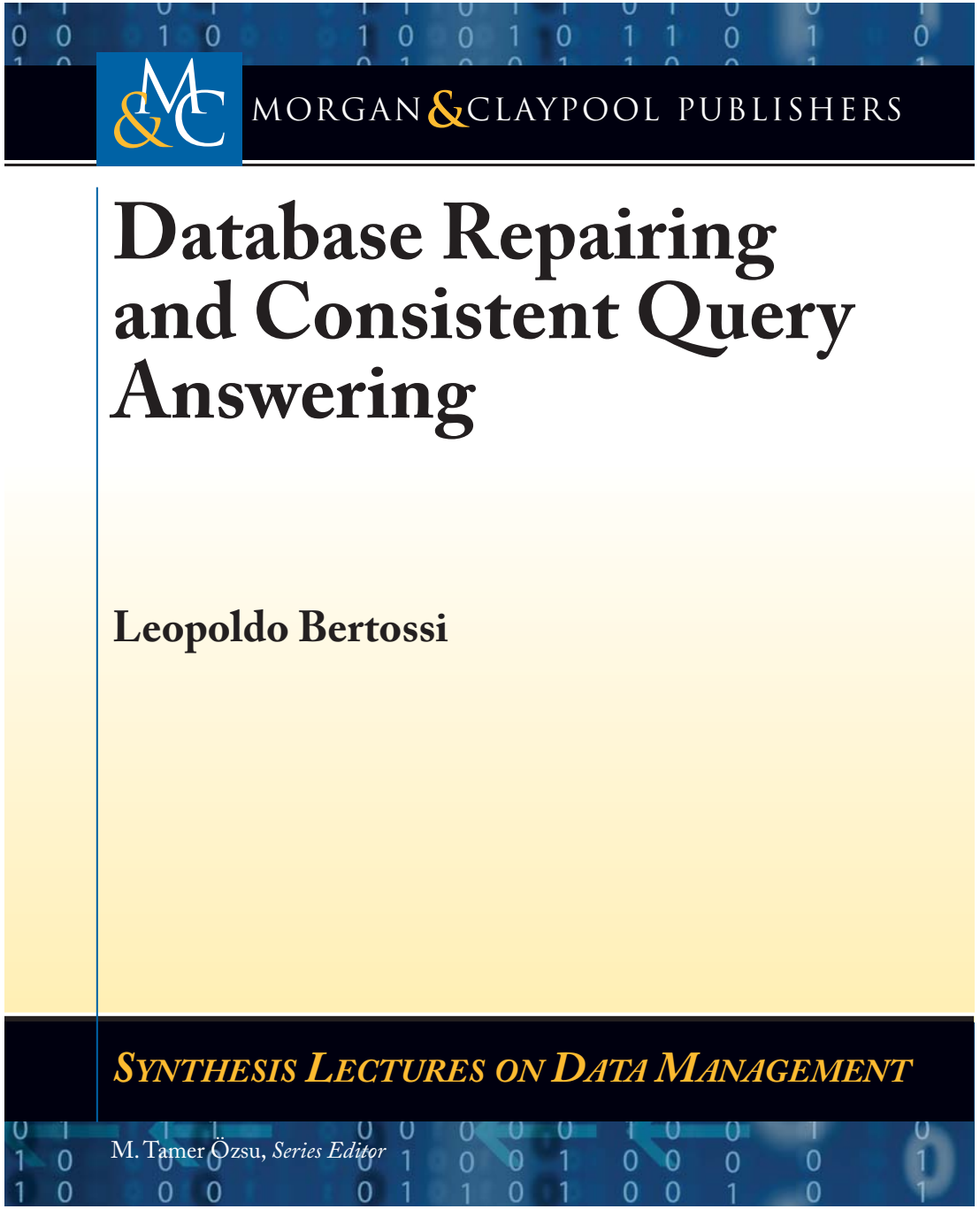
- Natural application scenario:

  Virtual data integration

  No way to enforce global ICs on the sources

  Inconsistencies have to be solved on-the-fly, at query-answering time




MORGAN & CLAYPOOL PUBLISHERS
Database Repairing and Consistent Query Answering
Leopoldo Bertossi
SYNTHESIS LECTURES ON DATA MANAGEMENT
M. Tamer Özsu, Series Editor

Conditional Dependencies (CDs)

Example: Database relation with FDs:

FD_1: $[CC, AC, Phone] \rightarrow [Street, City, Zip]$

FD_2: $[CC, AC] \rightarrow [City]$

CC	AC	Phone	Name	Street	City	Zip
44	131	1234567	mike	mayfield	NYC	EH4 8LE
44	131	3456789	rick	crichton	NYC	EH4 8LE
01	908	3456789	joe	mtn ave	NYC	07974

FDs are satisfied, but they are "global" ICs

They may not capture natural data quality requirements ...

... those related to specific data values

- What about a *conditional functional dependency* (CFD)?

$$CFD_1: \quad [CC = 44, Zip] \rightarrow [Street]$$

The FD of $Street$ upon Zip applies when the country code is 44

Not satisfied anymore, and data cleaning may be necessary ...

- More generally, CDs are like classical ICs with a *tableau* for forced data value associations

CFD_2:
$[CC = 44, AC = 131, Phone] \rightarrow [Street, City = \text{`}EDI', Zip]$

When $CC = 44, AC = 131$ hold, the FD of $Street$ and Zip upon $Phone$ applies, and the city is 'EDI'

Not satisfied either ...

- CQA and database repairs have been investigated for CFDs
- We can go one step further ...
- Conditional Inclusion Dependencies:

$$Order(Title, Price, Type = \text{`}book') \subseteq Book(Title, Price)$$

It can be expressed in classical FO predicate logic:

$$\forall x \forall y \forall z (Order(x, y, z) \wedge z = \text{`}book' \rightarrow Book(x, y))$$

Still a classic flavor ...

And semantics ...

Matching Dependencies (MDs)

- MDs are related to Entity Resolution (ER)
- ER is a classical, common and difficult problem in data cleaning

  ER is about discovering and merging records that represent the same entity in the application domain

  Detecting and getting rid of duplicates!
- Many *ad hoc* mechanisms have been proposed
- ER is fundamental for data analysis and decision making in BI
- Particularly crucial in data integration

- MDs express and generalize ER concerns

  They specify attribute values that have to be made equal under certain conditions of similarity for other attribute values

Example: Schema $R_1(X, Y), R_2(X, Y)$

$$\forall X_1 X_2 Y_1 Y_2 (R_1[X_1] \approx R_2[X_2] \longrightarrow R_1[Y_1] \doteq R_2[Y_2])$$

When the values for attributes X_1 in R_1 and X_2 in R_2 in two tuples are similar, then the values in those two tuples for attribute Y_1 in R_1 and Y_2 in R_2 must be made equal (matched)

(R_1 and R_2 can be same predicate)

$\approx$: Domain-dependent, attribute-level similarity relation

- MDs introduced by W. Fan et al. (PODS 2008, VLDB 2009)

- Although declarative, MDs have a procedural feel and a dynamic semantics

- An MD is satisfied by a pair of databases (D, D'):

  D satisfies the antecedent, and D', the consequent, where the matching (merging) is realized

  But this is local, one-step satisfaction ...

- We may need several steps until reaching an instance where all the intended mergings are realized

  Dirty instance: $D \Rightarrow D_1 \Rightarrow D_2 \Rightarrow \ldots\ldots \Rightarrow D'$

  $\uparrow$

  stable, clean instance!

  (there could be several of these)

  How each "$\Rightarrow$" step?

Matching Dependencies with MFs

"similar name and phone number ⇒ identical address"

D_0	*name*	*phone*	*address*
	John Doe	(613)123 4567	Main St., Ottawa
	J. Doe	123 4567	25 Main St.

⇓

D_1	*name*	*phone*	*address*
	John Doe	(613)123 4567	25 Main St., Ottawa
	J. Doe	123 4567	25 Main St., Ottawa

A dynamic semantics!

$$m_{address}(\underline{MainSt., Ottawa}\ ,\ \underline{25MainSt.}) := \underline{25MainSt., Ottawa}$$

Addresses treated as strings or objects, i.e. sets of pairs attribute/value

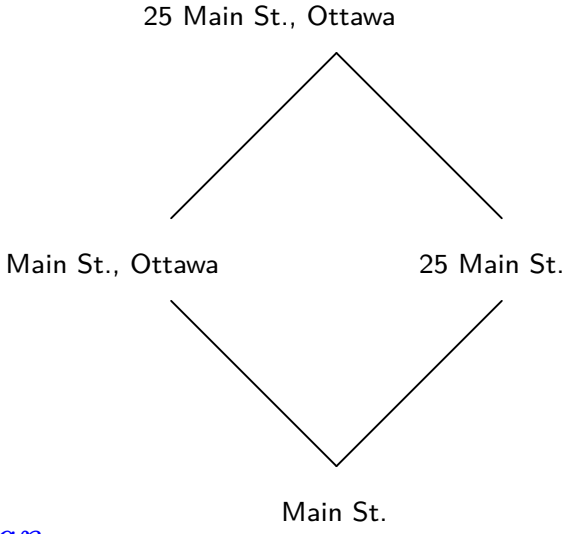
(Join work with Solmaz Kolahi and Laks Lakshmanan; ICDT'11, TOCS 2013)

- Matching functions induce a partial order (lattice) on attribute domains

$$a \preceq_A a' \iff m_A(a, a') = a'$$

$a \preceq_A a'$ can be thought of in terms of information contents

When MFs are applied we increase information contents, and decrease uncertainty!



$$D_0 \sqsubseteq D_1 \sqsubseteq \ldots \sqsubseteq D_{clean}$$

- In general, there could be multiple clean instances
- For two special cases:
  - Similarity-preserving matching functions

$$a \approx a' \Rightarrow a \approx \boldsymbol{m}_A(a', a'')$$

  - Interaction-free MDs

  There is a unique clean instance D_{clean}

  It can be computed in polynomial-time in data

ERBlox

Joint work with:

Zeinab Bahmani (Carleton University)

Nikolaos Vasiloglou (LogicBlox Inc.)

Entity Resolution

- A database may contain several representations of the same external entity

  The database has to be cleaned from duplicates

- The problem of entity resolution (ER) is about:

(A) Detecting duplicates, as pairs of records or clusters thereof

(B) Merging duplicates into single representations

- Much room for machine learning (ML) techniques

Blocking: Detecting Potential Duplicates

- We need to:

  (a) Compare pairs of records, for elements of a same entity (class):

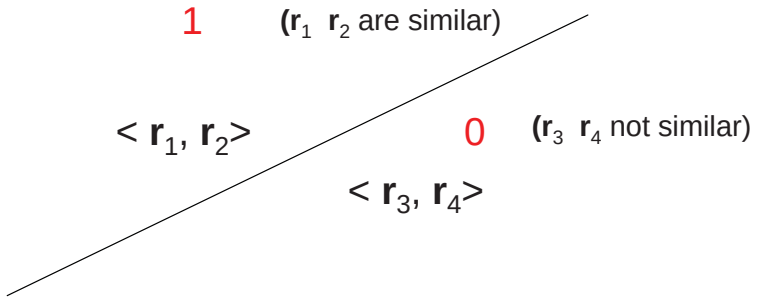
  $$\mathbf{r_1} = \langle a_1, \ldots, a_n \rangle \quad \text{vs.} \quad \mathbf{r_2} = \langle a'_1, \ldots, a'_n \rangle$$

  (b) Discriminate between pairs of duplicate records and pairs of non-duplicate records

- A classification problem

- In principle, every two records have to be compared, and classified

  This can be costly ...



- Need to reduce the large amount of two-record comparisons

  ER mechanisms use blocking techniques

- A single attribute in records, or a combination of attributes, called a blocking key, is used to split records into blocks

  $\mathbf{r} = \langle \underline{a_1}, a_2, \ldots, \underline{a_5}, \ldots, \underline{a_8}, a_9 \rangle$ BK $= \langle A_1, A_5, A_8 \rangle$

  Only records within the same block values are compared

  Any two records in different blocks will never be duplicates

- For example, block employee records according to the city

  Compare only employee records with the same city

- After blocking many record-pairs that are clear non-duplicates are not further considered

  But true duplicate pairs may be missed

- For example, due to data input errors or typographical variations in attribute values

  Even assuming data is free of those problems, we need "similarity" functions:

  "Joseph Doe" and "Joe Doe" may not be errors, but possible different representations of the same:

$$s_{name}(\text{“Joseph Doe”}, \text{“Joe Doe”}) = 0.9$$

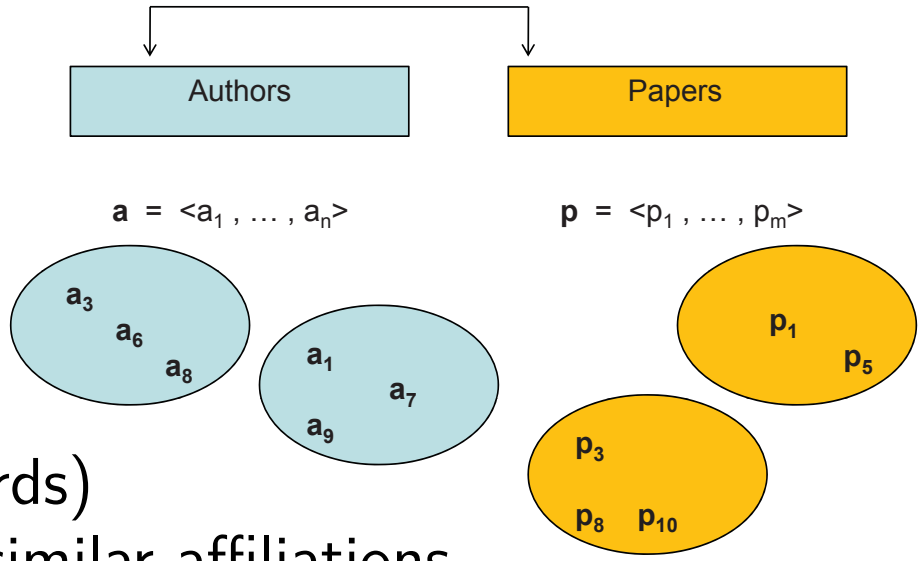
- So, now records in a same block have their BK attributes with "similar" values

- But still, grouping entities into blocks using just BK similarities may cause low recall

- It is useful to apply blocking with additional semantics and/or domain knowledge

Example: We have "author" and "paper" entities (records)

Want to group author entities based on similarities of authors' names and affiliations



Assume author entities
$\mathbf{a_1}, \mathbf{a_2}$ (complete author records)
have similar names, but not similar affiliations

$\mathbf{a_1}, \mathbf{a_2}$ are authors of papers (entities) $\mathbf{p_1}, \mathbf{p_2}$, resp.,

$\mathbf{p_1}, \mathbf{p_2}$ have been put in the same block of papers

Semantic knowledge: *"If two papers are in the same block, their authors with similar names should be in the same block"*

So, assign $\mathbf{a_1}, \mathbf{a_2}$ to same block (they could be duplicates)

- This is blocking of author and paper entities, separately, but collectively

  According to their relational closeness

  Not only on the basis of local similarities at the attribute level

- How can we capture this kind of additional semantic knowledge?

  With a MD like this:

$$Author(x_1, y_1, bl_1) \ \wedge \ Paper(y_1, z_1, bl_3) \ \wedge \ Author(x_2, y_2, bl_2) \wedge$$

$$Paper(y_2, z_2, bl_3) \ \wedge x_1 \approx_1 x_2 \ \wedge \ z_1 \approx_2 z_2 \ \longrightarrow \ bl_1 \doteq bl_2$$

  This is (an extended form of) a matching dependency (MD), used here for blocking

  Originally for merging attribute values, not for blocking

- ML could be used to create the blocks, e.g. using clustering methods (part of 1st ER phase)

  Not what we do here ...

- We use MDs for blocking, before the ML-based classification task

- Not quite clear how to develop ML-based classifiers involving semantic knowledge

  Some recent work on kernel-based methods with -assumed to be true- logical formulas and semi-supervised training

  (Diligenti et al., Machine Learning, 2012, 86(1):57-88)

- After blocking we may start classifying pairs

Classifying Record-Pairs (general)

- ML techniques are commonly used to discriminate between:
  - pairs of duplicate records (of each other), i.e. duplicate pairs, and
  - pairs of non-duplicate records, i.e. non-duplicate pairs
- ML is used here to classify record-pairs

  (still part of 1st phase of ER)
- We developed a classification model

  The classification hyper-plane in slide 25 ...

All this is part of the *ERBlox* approach/system

The *ERBlox* Approach to ER

- *ERBlox* enables/supports ML-techniques for ER

  Different ML techniques can be used for the classification model

  ER is based on supervised ML techniques, which require training data

  We used the "support-vector machine" (SVM) method (mainly)

- *ERBlox* also based on the use of MDs

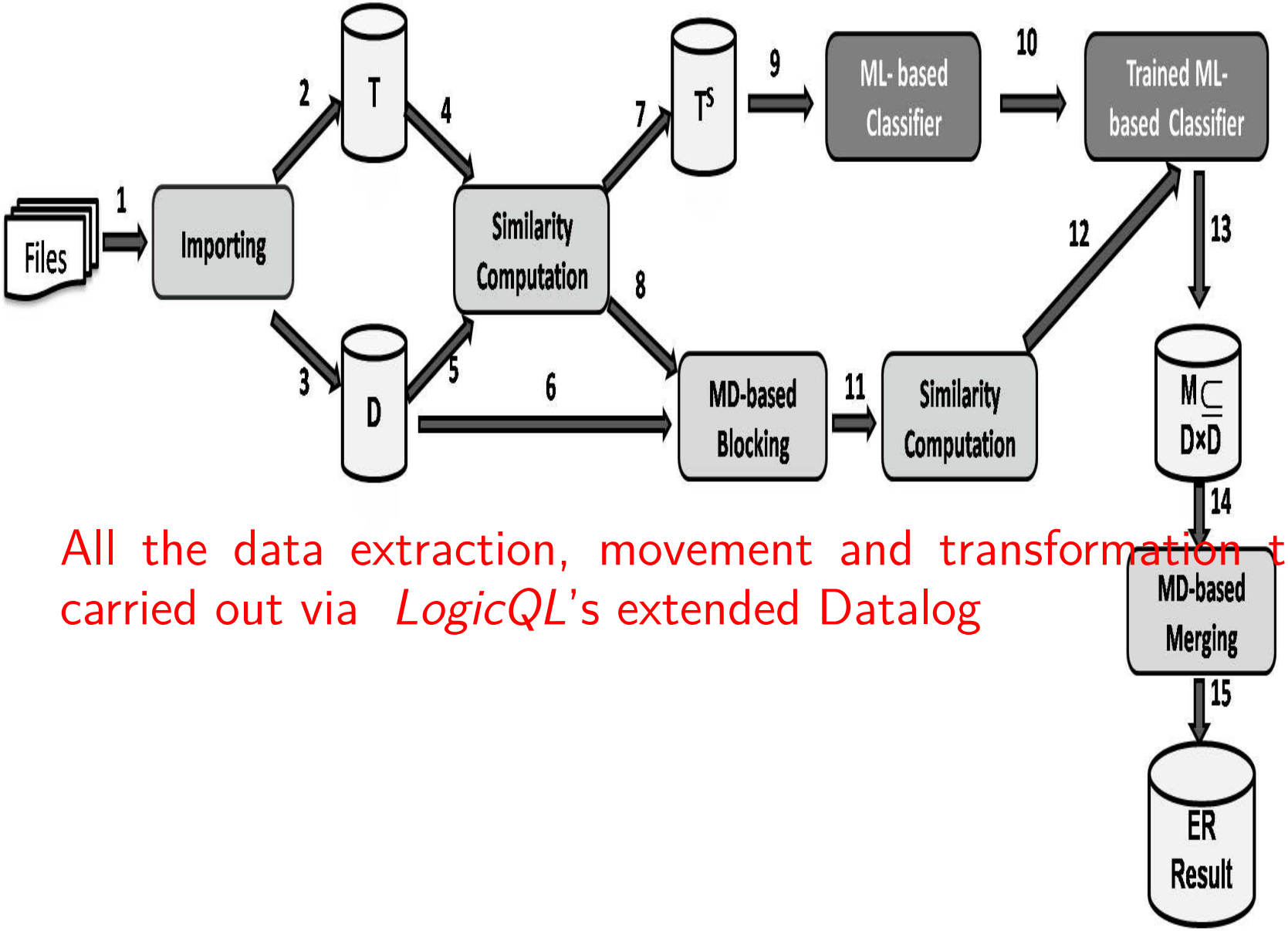
- *ERBlox* interacts with Datalog-based relational DBs

  Profiting from Datalog for different tasks

  More specifically, the *LogicBlox* system (*LogiQL*, now)

- *ERBlox* has three main components:

  1. MD-based collective blocking
  2. ML-based record duplicate detection
  3. MD-based merging



All the data extraction, movement and transformation tasks carried out via *LogicQL*'s extended Datalog

Merging

- After the classification task, (records in) duplicate-pairs have to be merged

  Records in them are considered to be "similar"

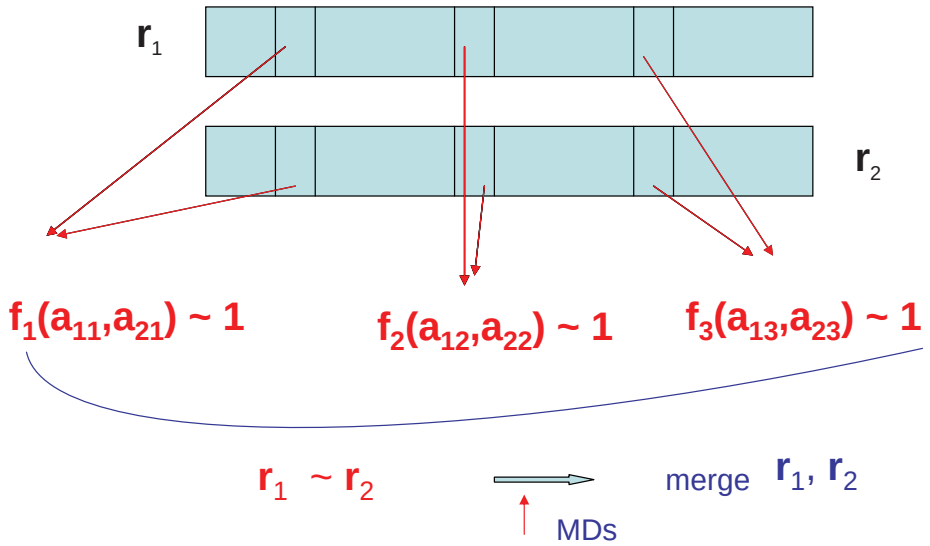
  In a precise mathematical sense, through the use of domain-dependent "features"

  MDs are also used for merging (their common use)

- Different sets of MDs for blocking and merging

- The classifier decides if records $\mathbf{r_1}, \mathbf{r_2}$ are duplicates

  In the positive case, by returning $\langle \mathbf{r_1}, \mathbf{r_2}, 1 \rangle$



- Define: $r_1 \sim r_2 \ :\Longleftrightarrow\ \langle r_1, r_2, 1\rangle$ is output
- Merge-MDs of the form: $r_1 \sim r_2 \ \rightarrow\ r_1 \doteq r_2$

  LHS means $\langle r_1, r_2\rangle$ is given value 1 by classifier

  RHS means $r_1[A_1] \doteq r_2[A_1] \wedge \cdots \wedge r_1[A_m] \doteq r_2[A_m]$
- Mergings on RHS, based on domain-dependent matching functions (MFs)

On the Use of MDs

- In general, application of MDs on an instance may produce alternative, admissible instances
- General MDs can be specified/enforced with answer-set programs (ASPs) [Bahmani et al., KR'12]

  General ASP not supported by *LogiQL*
- We obtain a single blocking solution, applying "blocking MDs"
- On that basis, also the final result of ER is a single duplicate-free instance, applying "merge-MDs"
- The kind of MDs in our case, and the way there are use/applied, requires only "stratified Datalog", which is supported by *LogiQL*

  Our MDs can be specified/executed with LogiQL's Datalog

MD-Based Collective Blocking

- Records have unique, global ids

  (positive integer values that can be compared with $<$)

  Initial block number $Bl\#$ for a record is its id

- Two records are forced to go into same block by enforcing the equality of their block numbers

  Use MDs with a MF: $m_{Bl\#}(b_i, b_j) := b_i$ if $b_j \leq b_i$

Example: Author and Paper entities ("$R.\ Smith$" $\approx$ "$MR.\ Smyth$")

$Author$	$Name$	$Affiliation$	$PaperID$	$Bl\#$
12	$R.\ Smith$	$MBA,\ UCLA$	1	12
13	$MR.\ Smyth$	MBA	2	13
14	$J.\ Doe$	$MBA,\ UCLA$	3	14

$Paper$	$Title$	$Year$	$AuthorID$	$Bl\#$
1	$Illness\ in\ Africa$	1990	12	2
2	$Illness\ in\ West\ Africa$	90	13	2

"Group two author entities into same block if they have similar names and affiliations or they have similar names and their corresponding papers are in same block"

$$m_1: \quad Author(a_1, x_1, y_1, p_1, b_1) \wedge Author(a_2, x_2, y_2, p_2, b_2) \wedge x_1 \approx x_2 \ \wedge \ y_1 \approx y_2 \ \rightarrow \ b_1 \doteq b_2$$

$$m_2: \quad Author(a_1, x_1, y_1, p_1, b_1) \ \wedge Author(a_2, x_2, y_2, p_2, b_2) \ \wedge x_1 \approx x_2 \wedge Paper(p_1, x_1', y_1', a_1, b_3) \ \wedge \ Paper(p_2, x_2', y_2', a_2, b_3) \ \rightarrow \ b_1 \doteq b_2$$

First is a single-entity blocking MD

The second is an inter-entity blocking MD

Applying them results in a DB instance with two author-blocks: $\{12, 13\}, \ \{14\}$

MD-Based Merging

- We enforce MDs to merge duplicate records into single representations
- We consider record-level MDs:

$$\varphi: \quad R[t_1] \approx R[t_2] \longrightarrow R[\bar{Z}_1] \doteq R[\bar{Z}_2]$$

$\bar{Z}_1$, $\bar{Z}_2$ contain all attributes of R

- The MDs implicitly contain all attributes on the LHS

  The LHSs imposes the condition that two tuples are duplicates (a higher-level notion of similarity)

  Its truth is evaluated according to the output of the classifier

Example: Merge duplicate author-records enforcing the MD:

$$Author[aid_1] \approx Author[aid_2] \longrightarrow$$
$$Author[Name, Affiliation, PaperID] \doteq Author[Name, Affiliation, PaperID]$$

(LHS abbreviation for $Author \sim Author$)

- A derived table $Author\text{-}Duplicate$ is used on LHS, with contents computed pre-merging and kept fixed during the enforcement of merge-MDs

  In this way, transitivity of record similarity is captured ...

  This makes the sets of merging-MDs interaction-free

  Resulting in a unique resolved instance
  (similarly for enforcement of blocking-MDs)

Experimental Evaluation

- We experimented with our *ERBlox* system using datasets of Microsoft Academic Search (MAS), DBLP and Cora

  MAS (as of January 2013) includes 250K authors and 2.5M papers, and a training set

- We used two other classification methods in addition to SVM

- The experimental results show that our system improves ER accuracy over traditional blocking techniques where just blocking-key similarities are used

- Actually, MD-based collective blocking leads to higher precision and recall on the given datasets

Final Remarks

- *ERBlox* developed in collaboration with the the LogicBlox company `http://www.logicblox.com/`

  It is built on top of the *LogicBlox* Datalog platform

- High-level goal is extend LogiQL

  Developed and used by LogicBlox

  They extend, implement and leverage Datalog technology

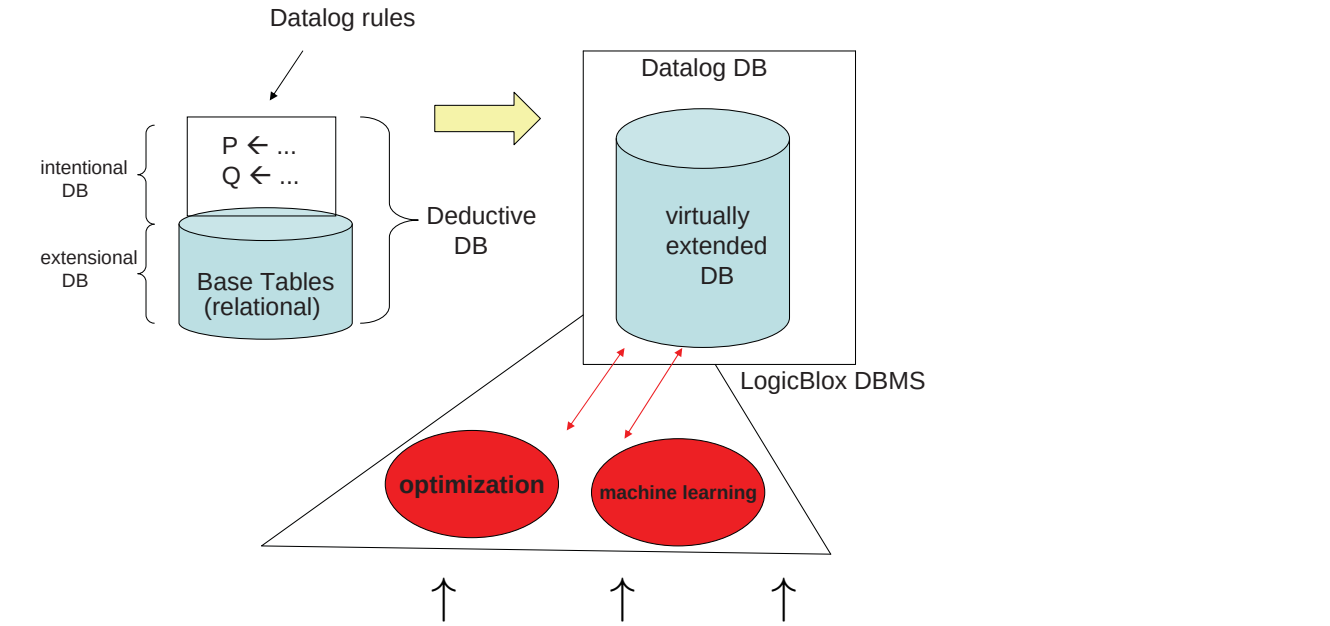
- Datalog has been around since the early 80s

  Used mostly in DB research

  It has experienced a revival during the last few years, and many new applications have been found!

- Datalog enables declarative and executable specifications of data-related domains

  An extension of relational algebra/calculus/databases



- LogicQL is being extended with interaction with optimization and machine learning packages and systems!

Data for these problems stored as "extensions" for DB & Datalog predicates

Optimization system reads necessary data from tables or Datalog computations

Optimization results may become contents for newly defined predicates

Smooth interaction between Datalog/relational engine and optimization/ML packages ...

- New ML methods are being added ...

  ML methods mostly developed in house

  `http://www.ismion.net/documentation/index.html`

Example: Optimize Shelf Space

total space product p takes on the shelf

```
totalShelf[]=u ← agg<<u=sum(z)>>
                    z=x*y, Stock[p] = x, spacePerProduct[p]=y
```

```
Product(p) → Stock[p] ≥ minStock[p]
Product(p) → Stock[p] ≤ maxStock[p]
totalShelf[]=u, maxShelf[]=v → u≤v
```

Integrity constraints: min/max/totalShelf

```
totalProfit[]=u ← agg<<u=sum(z)>>
                    z=x*y, Stock[p] = x, profitPerProd[p]=y
```

```
lang:solve:variable('Stock)
lang:solve:max('totalProfit)
```

unknown var

goal

total estimated profit

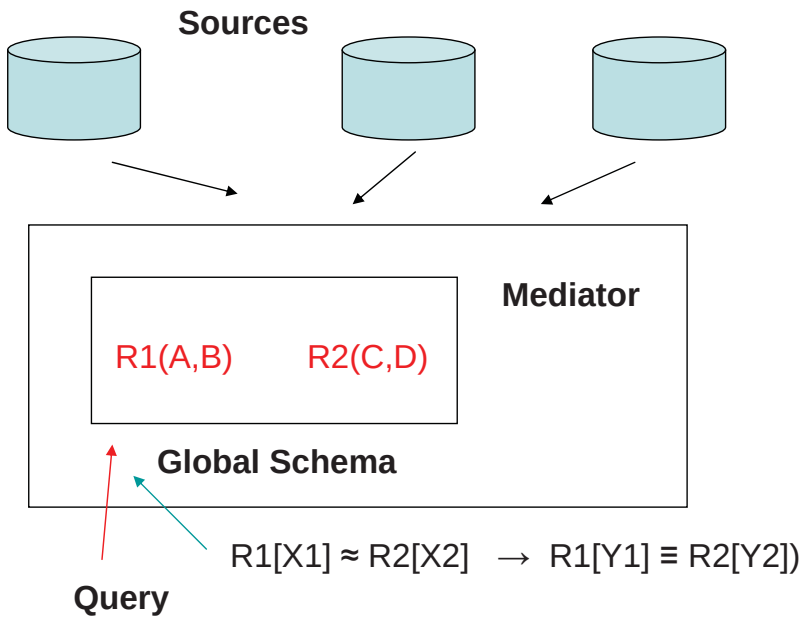
Interesting New Research Directions

- ER and virtual data integration

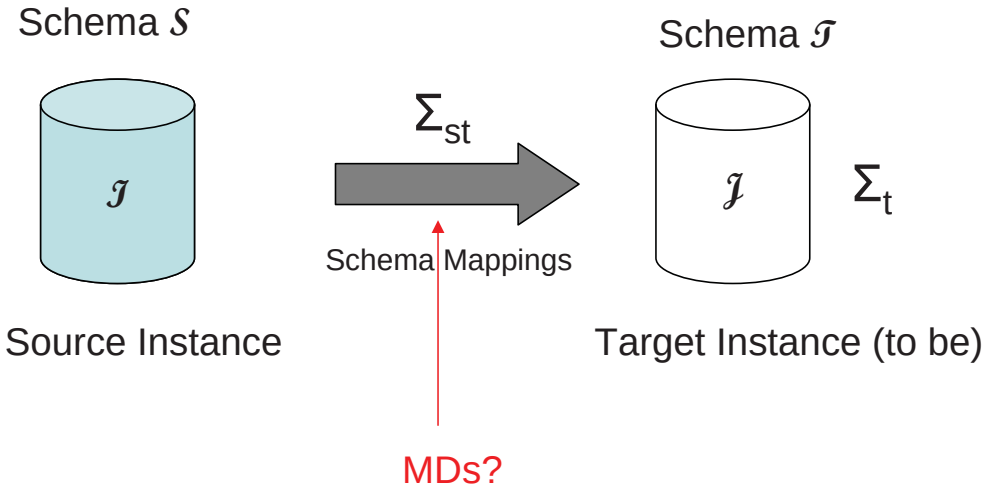
  Declarative specifications of ER could be compiled into query answering!

  Virtual data integration is a natural application scenario

  On-the-fly ER!



- ER and data exchange under schema mappings:



Traditionally: Materialize a (good) target instance $\mathcal{J}$ with:

$$(\mathcal{I}, \mathcal{J}) \models \Sigma_{st} \quad \text{and} \quad \mathcal{J} \models \Sigma_t$$

Now: also apply MDs when shipping data from $\mathcal{I}$ to $\mathcal{J}$

ER at data exchange time ...