ERBlox: Combining Matching Dependencies with Machine Learning for Entity Resolution

Leopoldo Bertossi

With: Zeinab Bahmani

Carleton University
School of Computer Science
Ottawa, Canada

Entity Resolution

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

  The database contains duplicate records, considered to be undesirable

  The database has to be cleaned ...

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

(A) Detecting duplicates, and

(B) Merging duplicates into single representations

- A classic and complex problem in data management, and data cleaning in particular

Starting ER: Detecting Potential Duplicates

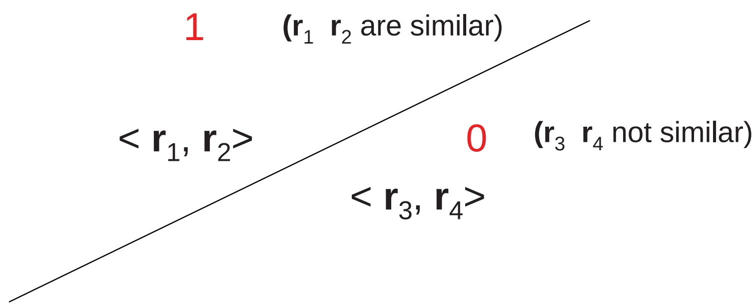
- We need to:

  (a) Compare pairs of records

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

- This becomes a classification problem

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



- To reduce the large amount of two-record comparisons, most ER systems use blocking techniques

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

  Only records within the same block are compared

  Any two records in different blocks will never be duplicates

- For example, block a employee records according to the city

  We compare only employees 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

  We assume data is free of this kind of problems

  Still introducing "similarity" functions becomes necessary:

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

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

- But still, grouping the entities into blocks based just on blocking-key similarities may cause low recall

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

Example: 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) p_1, p_2, resp., which 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

Considering this, we may group $\mathbf{a}_1, \mathbf{a}_2$ in the same block

We would be blocking author and paper entities, separately, but collectively

... according to their relational closeness (not only based of local similarities at attribute level)

How can we capture this kind of additional knowledge?

Informally, with something like this:

$$Author(x_1, y_1, bl_1) \wedge Paper(y_1, z_1, bl_3) \wedge Author(x_2, y_2, bl_2) \wedge$$
$$Paper(y_2, z_2, bl_3) \wedge x_1 \approx_1 x_2 \wedge z_1 \approx_2 z_2 \ \longrightarrow \ bl_1 \doteq bl_2$$

- We use matching dependencies (MDs) for blocking

Classifying Records

- Machine learning (ML) techniques are commonly used to discriminate between pairs of duplicate and non-duplicate records (after blocking)

  Some pairs are considered to contain two duplicates (of each other), and other pairs to contain non-duplicates

- ML is being used here to classify record-pairs (problem (b) in slide 3)

  ML could be used to create the blocks, e.g. using clustering methods (problem (a) in slide 3)

  Not what we do here ...

- We develop a classification model (coming)

  The classification hyper-plane in slide 3 ...

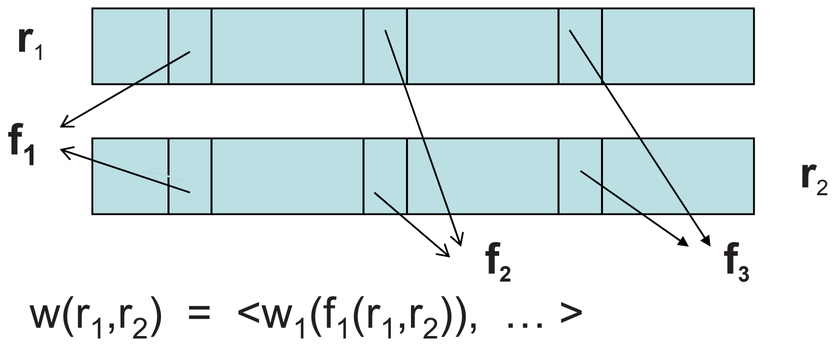
- We used MDs for blocking, before the ML task
- Not clear how to develop ML-based classifier involving semantic knowledge

  There is some recent work on kernel-based methods that use (assumed to be true) logical formulas together with semi-supervised training[1]

- Most of the work on applying ML to ER do things at the record level

  However, only some of the attributes, or their features, as we will see, may be involved detection (task (A) in slide 2)

[1] Cf. http://people.scs.carleton.ca/~bertossi/trabajo/learningWconstraintsPUC14.pdf



The f_i are real-valued functions of pairs of attribute values

The weight functions w_i assign a similarity values in, say [0,1]

- The choice of relevant sets of attributes and features is application/domain dependent

***ERBlox*'s Context**

- System 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

  Developed to extend, implement and leverage Datalog technology

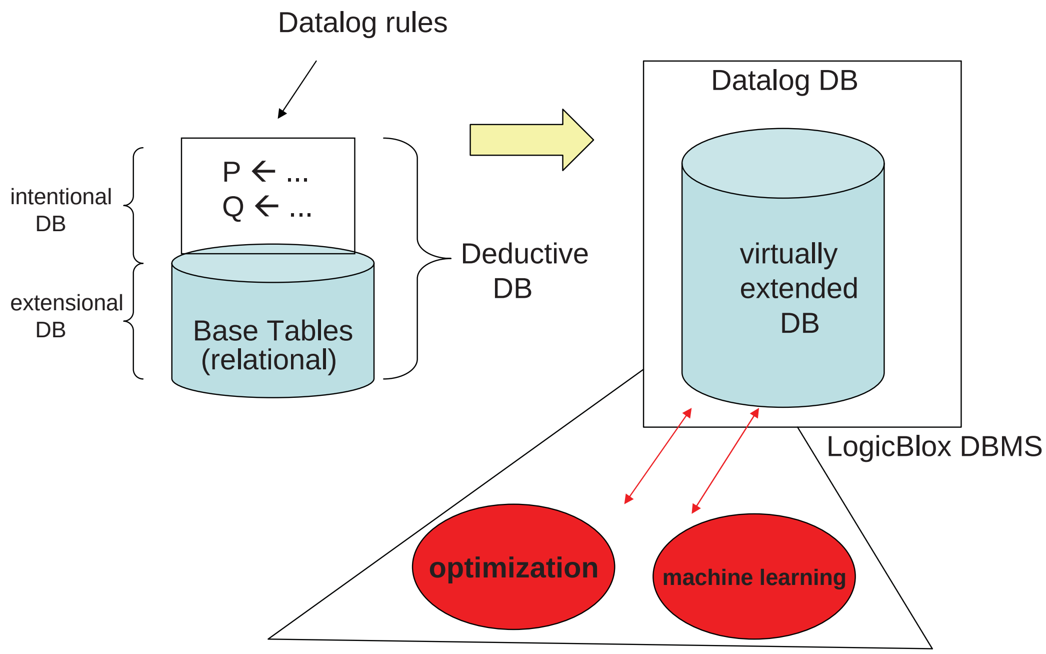
- Datalog has been around since the early 80s

  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

Optimizer reads necessary data from tables or Datalog computations

Results of optimizations may become contents for newly defined predicates

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

- New optimization and ML methods are being added ...

Optimization methods/packages developed by other groups and companies

ML methods mostly developed in house ...

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

ERBlox

- *ERBlox*'s development fits into the general goal

  Enabling ML-techniques for data cleaning

- *ERBlox*'s approach to- and implementation of ER uses ML and MD-based techniques

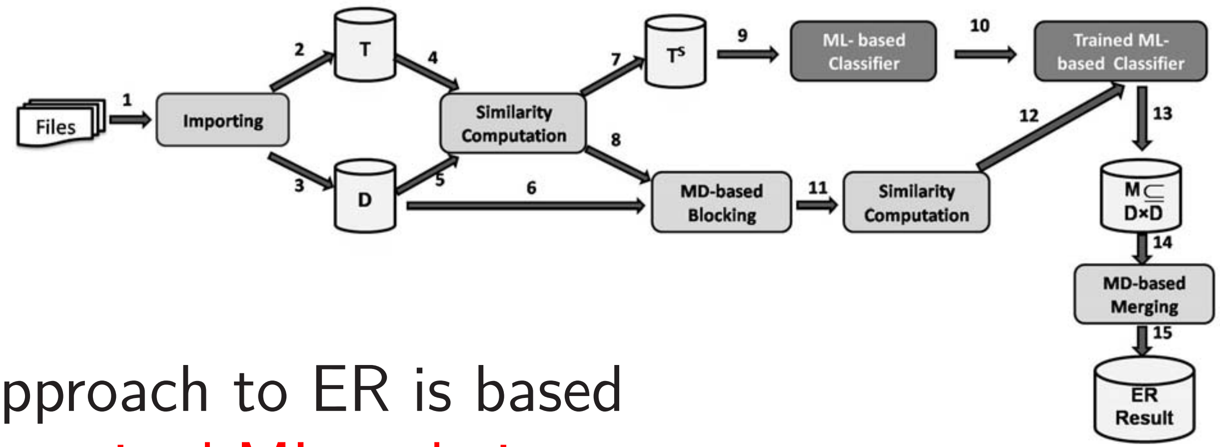
- *ERBlox* allows for the interaction of Datalog, MDs, and supervised ML techniques for ER

- *ERBlox* contains three main components:

  1. MD-based collective blocking

  2. ML-based record duplicate detection

  3. MD-based merging



- Our approach to ER is based on supervised ML techniques, which require training data

- We used the "support-vector machine" (SVM) method to produce a classification model

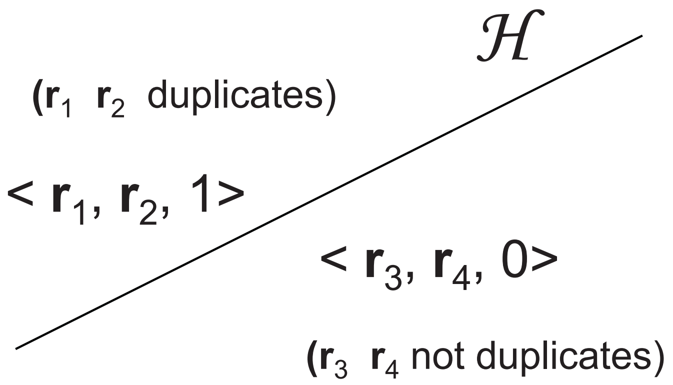
  In the end it is used to identify pairs of similar records

A bit of SVM:

- SVMs technique a form of kernel-based learning
- SVMs can be used for classifying vectors in an inner-product vector space $\mathcal{V}$ over $\mathbb{R}$
- Vectors are classified in two classes, with a label in $\{0, 1\}$
- The algorithm learns from a training set, say:

  $\{(\mathbf{e}_1, f(\mathbf{e}_1)), (\mathbf{e}_2,\ f(\mathbf{e}_2)), (\mathbf{e}_3, f(\mathbf{e}_3)),\ \ldots, (\mathbf{e}_n,\ f(\mathbf{e}_n))\}$

  $\mathbf{e}_i \in \mathcal{V}$, and for the *feature* (function) f: $f(\mathbf{e}_i) \in \{0, 1\}$
- SVMs find an optimal hyperplane, $\mathcal{H}$, in $\mathcal{V}$ that separates the two classes where the training vectors are classified



- Hyperplane $\mathcal{H}$ has equation of the form $\mathbf{w} \bullet \mathbf{x} + b = 0$
  - $\bullet$ is inner product
  - $\mathbf{x}$ is a vector variable; $\mathbf{w}$ a weight vector of real values
  - b is a real number

- New vector $\mathbf{e}$ in $\mathcal{V}$ can be classified as positive or negative depending on the side of $\mathcal{H}$ it lies

  Determined by computing $h(\mathbf{e}) := sign(\mathbf{w} \bullet \mathbf{e} + b)$

  If $h(\mathbf{e}) > 0$, $\mathbf{e}$ belongs to class 1; otherwise, to class 0

- It is possible to compute real numbers $\alpha_1, \ldots, \alpha_n$, such that:

  $$h(\mathbf{e}) = sign(\sum_i \alpha_i \cdot f(\mathbf{e}_i) \cdot \mathbf{e}_i \bullet \mathbf{e} + b)$$

- On the basis of the detected similarities, duplicates have to be merged

  For this, MDs are used again

- MDs are used for two tasks (common use is 3.)

- The sets of MDs are different for blocking and merging ...

Interlude: Matching Dependencies

Example: Relational schema

$R(X)$, $S(Y)$, $\quad X_1, X_2 \subseteq X, \;\; Y_1, Y_2 \subseteq Y, \;\; |X_1| = |Y_1|, \;\; |X_2| = |Y_2|$

$$\varphi:\ R[X_1] \approx S[Y_1] \quad \longrightarrow \quad R[X_2] \doteq S[Y_2]$$

"*If in two tuples of R, S, resp., the values for attribute(s) X_1, Y_1 are similar, the values in them for attribute(s) X_2, Y_2 must be matched/merged, i.e. made equal*"

R and S could be the same, and $X_2 = Y_2 = X = Y$; $\quad \approx$ is domain-dependent *"similar name and phone number $\Rightarrow$ identical address"*

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

$\Longrightarrow$

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

Matching function:

$\mathfrak{m}_{address}$('Main St., Ottawa', '25 Main St.') := '25 Main St., Ottawa'

- MDs provide a declarative language with a precise semantics could be used for merging duplicate records
- Matching Dependencies (MDs) were proposed

(Fan et al., PODS'08, VLDB'09)

- They are rules for resolving pairs of duplicate representations
- Previous work on ER via MDs have been concentrated mostly on introduction of MDs and the merging part of the ER problem via MDs

  We went beyond ...

MDs with Matching Functions:

- MDs have a dynamic semantics: $(D_0, D_1) \models \varphi$

  How to do the matching?

- Revised semantics for MDs: [Bertossi et al., ICDT'11, TOCS 2013]

$$\varphi: \quad R_1[\bar{X}_1] \approx R_2[\bar{X}_2] \ \rightarrow \ R_1[A_1] \doteq R_2[A_2]$$

  - $(D, D') \models \varphi$ if for every R_1-tuple t_1 and R_2-tuple t_2:

$$t_1[\bar{X}_1] \approx t_2[\bar{X}_2], \text{ but } t_1[A_1] = a_1 \neq t_2[A_2] = a_2 \text{ in } D$$
$$\Longrightarrow \quad t_1[A_1] = t_2[A_2] = \boldsymbol{m}_A(a_1, a_2) \text{ in } D'$$

  - D' is stable if $(D', D') \models \Sigma$ (a set of MDs)

- Chase procedure:

$$D \;\Rightarrow_{\varphi_1} D_1 \;\Rightarrow_{\varphi_2} D_2 \;\Rightarrow_{\varphi_3} \;\cdots\cdots\; \Rightarrow_{\varphi_n} \; D'$$

dirty instance stable (clean) instance

- Matching functions (MFs): $\mathfrak{m}_A(a_1, a_2)$
  - Attribute-domain dependent, commutative and associative
  - Induces a semilattice on domain A with partial order defined by

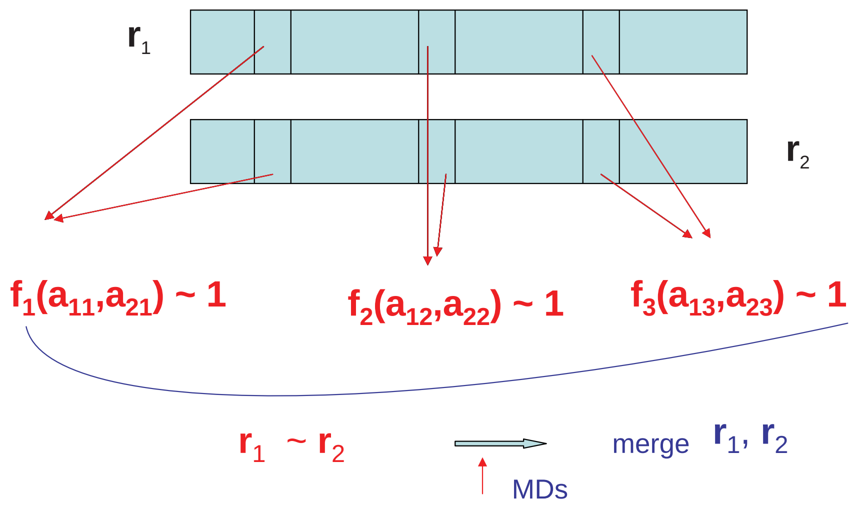
$$a \preceq_A a' \;:\Leftrightarrow\; \mathfrak{m}_A(a, a') = a'$$

  - LUB operator coincides with matching function:

$$lub\{a, a'\} = \mathfrak{m}_A(a, a') \quad \text{and} \quad a, a' \preceq_A \mathfrak{m}_A(a, a')$$

- It can be proved that for "interaction-free" there is a single resolved instance and can be computed in polynomial-time in data

Merging:



- In any case, it is the classifier that decides if r_1, r_2 are duplicates

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

- Define: $r_1 \sim r_2 \ :\Longleftrightarrow\ \langle r_1, r_2, 1\rangle$ is output

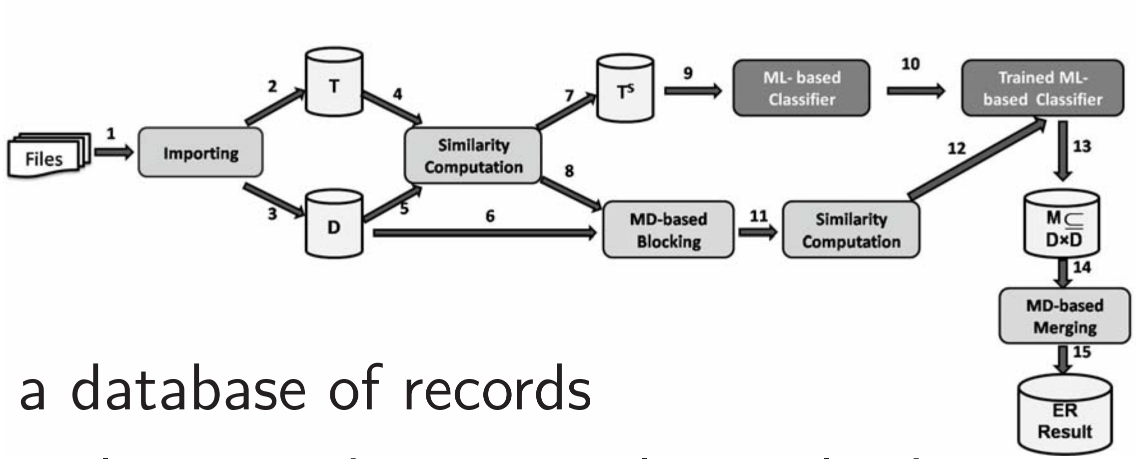
- Merge-MDs of the form: $r_1 \sim r_2 \ \rightarrow\ r_1 \doteq r_2$

  LHS means $\langle r_1, r_2\rangle$ is given value 1 by ML-based model

  RHS means $r_1[A_1] \doteq r_2[A_1] \wedge \cdots \wedge r_1[A_m] \doteq r_2[A_m]$

  (each record with exactly the m attributes A_i)

ERBlox Revisited



1. D is a database of records

   Say, tuples over the same relational schema

   Each entity has (is) a table; each row a record

2. Records are divided in blocks

   "Similarity Computation" generates similarities required for blocking

3. On the basis of computed blocks, similarities within records pairs are computed, again "Similarity Computation"

   Features for some attributes are used here

4. ML technique is trained using set of training examples T

   T^s: set of labeled record-pairs of records, with 1 or 0 (duplicates/non-duplicates)

   (Labels consistent with the values provided by the chosen features)

   T^s is used by the ML technique as a basis for developing the learning algorithm, the classification model

5. The classification model is applied to all record-pairs, with records in a pair coming from same block

   Any two full records (in the same record-pair) are declared as duplicates or non-duplicates

   In the former case, they will be fully merged

6. M is a DB of output record-pairs labeled with 1

   Their two records will be merged

7. ER result is obtained as a duplicate-free instance by applying *merge-MDs* to M

- General MDs can be implemented/specified with answer-set programs (ASPs) [Bahmani et al., KR'12]

  General ASP not supported by *LogiQL*

- The kind of MDs in our case requires only "stratified Datalog", which is supported by *LogiQL*

  LogiQL is used to specify and implement the enforcement of MDs, both for the blocking and merge steps

- By syntax of set of blocking-MDs and enforcement method of merge-MDs (using auxiliary data structures), both sets of MDs turn out to be interaction-free: single solutions computable in polynomial time!

Some Specifics about *ERBlox*

Similarity Computation:

- For record blocking, similarity measures are needed, to decide if two records r_1, r_2 go to the same block

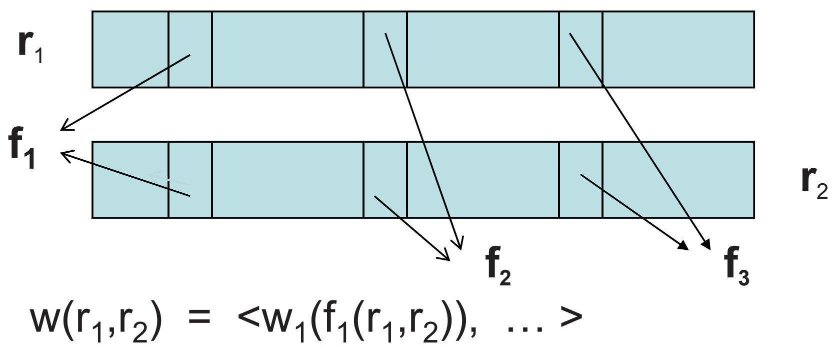
  For record-pairs $\langle r_1, r_2 \rangle$ in T, similarities have to be computed

- Similarity computation based on similarity functions

  $Sf_i : Dom_{A_i} \times Dom_{A_i} \rightarrow [0, 1]$

- Record-pairs get weight vectors

  $w(r_1, r_2) = \langle \cdots, Sf_i(r_1[A_i], r_2[A_i]), \cdots \rangle$



- Training set T leads to T^s of tuples $\langle r_1, r_2, w(r_1, r_2), L \rangle$, with $L \in \{0, 1\}$

MD-Based Collective Blocking

- MD-based collective blocking is a novel blocking technique

  It applies semantics or domain knowledge for effective blocking

  In contrast to most blocking techniques, MD-based collective blocking extends blocking, beyond the use of just similarity, by capturing relational closeness

- In D records have unique, global tids (positive integer values, can be compared with $<$)

  Initially assign a blocking number, $Bl\#$, to each record, initially the tid value

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

  Use MDs with matching functions: $m_{Bl\#}(t_i, t_j) = t_i$ if $t_j \leq t_i$

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”

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

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

MDs at the attribute level, as usual

The attributes corresponding to the chosen features (for blocking, possibly different than for classification) appear on LHSs of MDs

Applying the MDs (1), (2) results in an instance and the set of author blocks $\{\{12, 13\}, \{14\}\}$

Record Duplicate Detection via SVMs

- First the SVM classifier trained with T^s containing tuples of the form $\langle r_1, r_2, w(r_1, r_2)\rangle$

  $w(r_1, r_2)$: computed weight vector for records (with ids) r_1, r_2 in a same block

  (Other ML-classification methods can be invoked from *LogiQL* through a generic Datalog interface)

- The classification model is computed, as a separating hyperplane

- The input to classifier is the set of (extended) record-pairs $\langle r_1, r_2, w(r_1, r_2)\rangle$

- The output is a set of record-pairs $\langle r_1, r_2, 1\rangle$ or $\langle r_1, r_2, 0\rangle$

- If $\langle \ldots, 1\rangle$ and entity is R, a tuple $R\text{-}Duplicate(r_1, r_2)$ is created, to be used with the *LogicQL* program for MD-based merging

Example: (cont.) Consider the blocks for entity Author

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

Attributes $Name$ and $Affiliation$ are used for feature-based weight vector computation

With Author records, input to classifier: $\langle 12,\ 13, w(12,13)\rangle$, with $w(12,13) = [0.8, 0.3]$

(Real input to the trained SVMs, is $[0.8, 0.3]$)

The SVM-classifier returns $\langle [0.8, 0.3], 1\rangle$, and system creates $AuthorDuplicate(12, 13)$

MD-Based Merging

- We enforce MDs to merge duplicate records into a single representations

- The MDs implicitly contain all attributes on the LHS

  The LHSs indicates that if two tuples are duplicates (a higher-level notion of similarity) are evaluated using the output of the classifier

  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

Example: Merge duplicate Author records enforcing the MD:

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

Derived table *Author-Duplicate* is used on LHS, with contents computed before and kept fixed during the enforcement of this merge-MD

Transitivity of record similarity is captured ...

This also has the effect of making the set of merging-MDs interaction-free

Resulting in a unique resolved instance

A stratified Datalog program is expressive enough for specifying and enforcing MD-based merging

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