Dutch-Belgian Database Day
– Past and Present of Entity Resolution

Keynote Information

date: Tuesday, 7 December 2021, 13:00-13:45
presenter: Ekaterini Ioannou http://www.eioannou.nl

Department of Management, University of Tilburg
• Introduction
• Time evolution of challenges
• Generations based on the challenges
• Latest developments & directions

Presentation based on
• Encode a large part of our knowledge
• Valuable asset for numerous current applications and (Web) systems
• Many names, descriptions, or IDs (URIs) are used for the same real-world objects

• Example:

  London
  लंडन
  لندن
  Llundain
  Londain
  Londe
  Londen
  Londen
  Lunden
  Lundúnir
  Lunnon
  London
  Londona
  Londonas
  Londoni
  Londono
  Londra
  Londres
  Londrez
  Londyn
  Lontoo
  Loundres
  Luân Đôn
  لندن
  لندن
  لندن
  Londinium
  London Eye
  Westminster Abbey
  future host of the XXX Olympic Games
  city of the Westminster Abbey
  city of the London Eye
  the city described by Charles Dickens in his novels

  http://sws.geonames.org/2643743/
  …
B. Disambiguation, Deduplication, etc.

• Plethora of **different objects** have the same name

• Examples:

  - London, KY
  - London, Laurel, KY
  - London, OH
  - London, Madison, OH
  - London, AR
  - London, Pope, AR
  - London, TX
  - London, Kimble, TX
  - London, MO

  - London, Jack
    2612 Almes Dr
    Montgomery, AL
    (334) 272-7005

  - London, Jack R
    2511 Winchester Rd
    Montgomery, AL 36106-3327
    (334) 272-7005

  - London, Jack
    1222 Whitetail Trl
    Van Buren, AR 72956-7368
    (479) 474-4136

  - London, Jack
    7400 Vista Del Mar Ave
    La Jolla, CA 92037-4954
    (858) 456-1850

  - …
Today’s situation

• Content providers offering valuable information describing (part of) real-world objects
• Information is useful for data integration, link discovery, query processing, searching, etc.

News about London
Wiki pages about the London
Social networks in London
Videos and tags for London
Reviews on hotels in London
Pictures and tags about London
Entity Resolution

- Task that identifies and aggregates the different profiles that describe the same real-world objects [1, 2, 3, 4, 5]
- Primary usefulness:
  - Improves data quality and integrity
  - Fosters re-use of existing data sources
- Example application domains:
  - Linked Data
  - Building Knowledge Graphs
  - Census data
  - Price comparison portals
Data collections can be of two types: clean + dirty [3, 5, 6]

1. Clean:
   - Duplicate-free data
   - E.g., DBLP, ACM Digital Library, Wikipedia, Freebase

2. Dirty:
   - Contain duplicate profiles
   - E.g., Google Scholar, CiteseerX
• Based on the quality of input, we distinguish entity resolution into 3 sub-tasks:

1. Clean-Clean ER, a.k.a. *Record Linkage* in databases
2. Dirty-Clean ER
3. Dirty-Dirty ER  
   equivalent to *Dirty ER*, a.k.a. *Deduplication* in databases


• Introduction
• Time evolution of challenges
• Generations based on the challenges
• Latest developments & directions
## Overview of Challenges!

<table>
<thead>
<tr>
<th>title</th>
<th>A Blocking Framework for ER …</th>
</tr>
</thead>
<tbody>
<tr>
<td>conference</td>
<td>TKDE 2013</td>
</tr>
<tr>
<td>author</td>
<td>George Papadakis</td>
</tr>
<tr>
<td>author</td>
<td>Ekaterini Ioannou</td>
</tr>
<tr>
<td>author</td>
<td>Themis Palpanas</td>
</tr>
<tr>
<td>author</td>
<td>Claudia Niederée</td>
</tr>
<tr>
<td>author</td>
<td>Wolfgang Nejdl</td>
</tr>
<tr>
<td>title</td>
<td>A Blocking Framework for ER …</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>conference</td>
<td>TKDE 2013</td>
</tr>
<tr>
<td>author</td>
<td>G. Papadakis</td>
</tr>
<tr>
<td>author</td>
<td>E. Ioannou</td>
</tr>
<tr>
<td>author</td>
<td>T. Palpanas</td>
</tr>
<tr>
<td>author</td>
<td>C. Niederée</td>
</tr>
<tr>
<td>author</td>
<td>W. Nejdl</td>
</tr>
</tbody>
</table>

**Overview of Challenges!**

**heterogeneity:** abbreviations, misspellings, acronym, initials, etc.
<table>
<thead>
<tr>
<th>title</th>
<th>A Blocking Framework for ER …</th>
</tr>
</thead>
<tbody>
<tr>
<td>conference</td>
<td>TKDE 2013</td>
</tr>
<tr>
<td>author</td>
<td>G. Papadakis</td>
</tr>
<tr>
<td>author</td>
<td>E. Ioannou</td>
</tr>
<tr>
<td>author</td>
<td>T. Palpanas</td>
</tr>
<tr>
<td>author</td>
<td>C. Niederée</td>
</tr>
<tr>
<td>author</td>
<td>W. Nejdl</td>
</tr>
<tr>
<td>publication</td>
<td>A Blocking Framework for ER …</td>
</tr>
<tr>
<td>venue</td>
<td>TKDE 2013</td>
</tr>
<tr>
<td>researcher</td>
<td>George Papadakis</td>
</tr>
<tr>
<td>researcher</td>
<td>Ekaterini Ioannou</td>
</tr>
<tr>
<td>researcher</td>
<td>Themis Palpanas</td>
</tr>
<tr>
<td>researcher</td>
<td>Claudia Niederée</td>
</tr>
<tr>
<td>researcher</td>
<td>Wolfgang Nejdl</td>
</tr>
</tbody>
</table>

Overview of Challenges!

heterogeneity, schema variations
### Overview of Challenges!

- **heterogeneity, schema variations, collective**

| title | A Blocking Framework for ER … |
| conference | TKDE 2013 |
| author | G. Papadakis |
| author | E. Ioannou |
| author | T. Palpanas |
| author | C. Niederée |
| author | W. Nejdl |

→ alternatives matches increase the belief

| title | Entity Resolution: Past, Present and Yet-to-Come |
| conference | EDBT 2020 |
| author | George Papadakis |
| author | Ekaterini Ioannou |
| author | Themis Palpanas |
| author | Claudia Niederée |
| author | Wolfgang Nejdl |
Overview of Challenges!

→ Propagate information of a detected match

heterogeneity, schema variations, collective
Overview of Challenges!

- heterogeneity
- schema variations
- collective
- unstructured
heterogeneity, schema variations, collective, unstructured, volatility
<table>
<thead>
<tr>
<th>Topic</th>
<th>Year</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Blocking Framework</td>
<td>TKDE 2013</td>
<td>Papadakis, George, Ioannou, Palpanas</td>
</tr>
</tbody>
</table>

heterogeneity, schema variations, collective, unstructured, volatility, Big collections, …
Overview of Challenges!

heterogeneity, schema variations, collective, unstructured, volatility, Big collection,
Challenges

• ER challenges arise from the application settings

• Examples:
  ◦ Data characteristics
  ◦ System and resources
  ◦ Time restrictions
  ◦ …

• Evolving nature of the application settings implies:
  ◦ Constant modification of the challenges
  ◦ Plethora of resolution methods
• Introduction
• Time evolution of challenges
• Generations based on the challenges
• Latest developments & directions

additional material at the end of the presentation
Veracity
+Volume
+Variety
+Velocity

ER Methods

- Data Evolution
- Schema Diversity
- Profile Noise
- Source Size
• Scope → Veracity

  ◦ **Structured data** with known semantics and quality, e.g., small relational databases
  ◦ Dealing with high levels of profile noise

• Goal:

  ◦ Achieve high accuracy despite inconsistencies, noise, or errors in profiles

• Assumption is a “known schema”
• Creating mappings between equivalent attributes of the two schemata, e.g., profession ≡ job

• Grouping similar profiles into blocks
  ◦ All profiles in one block might be the same
  ◦ Profiles of different blocks cannot be the same

• Estimating the similarity among the candidate matches

• Partitioning the matched pairs into equivalence clusters, i.e., sets describing the same real-world object
Veracity
+ Volume
+ Variety
+ Velocity

ER Methods
• **Scope → Volume & Veracity**
  ◦ *(tens of)* millions of structured profiles

• **Goals:**
  ◦ High accuracy despite noise
  ◦ High time efficiency despite the size of data

• **Assumptions:**
  ◦ Known schema → custom, schema-based solutions

• **Workflow remains the same**
Solution: Parallelization

- **Type A :: Multi-core parallelization**
  - Single system → shared memory
  - Distribute processing among available CPUs

- **Type B :: Massive parallelization**
  - Cluster of independent systems
  - Map-Reduce paradigm [1]
    - Data partitioned across the nodes of a cluster
    - **Map Phase**: transforms a data partition into (key, value) pairs
    - **Reduce Phase**: processes pairs with the same key

- **Additional material (end of presentation)**
  - Parallelization method for each workflow step
Veracity
+Volume
+Variety
+Velocity

ER Methods
Scope → Variety & Volume & Veracity

- **User-generated Web Data**
- Users are free to add attribute values and/or attribute names
  → unprecedented levels of schema heterogeneity
  - Google Base: 100,000 schemata for 10,000 profile types
  - BTC09: 136,000 attribute names
- **Voluminous, (semi-)structured datasets**
  - BTC09: 1.15 billion triples, 182 million profiles
- Several datasets produced by automatic information extraction techniques → noise, tag-style values

(BTC: billion triple challenge database)
Example of Web Data

DATASET 1

Entity 1
- name=United Nations Children’s Fund
- acronym=unicef
- headquarters=California
- address=Los Angeles, 91335

Entity 2
- name=Ann Veneman
- position=unicef
- address=California
- ZipCode=90210

DATASET 2

Entity 3
- organization=unicef
- California
- status=active
- Los Angeles, 91335

Entity 4
- firstName=Ann
- lastName=Veneman
- residence=California
- zip_code=90201

Loose Schema Binding
Split values
Attribute Heterogeneity
Noise
• Schema Matching → not applicable (too many alternatives)
• Instead, partition attributes according to their syntactic similarity, regardless of their semantic relation
• Goal: Facilitate next steps
• Both Clean-Clean and Dirty ER
• Attribute clustering using graphs
• Considers all attribute values and completely ignores all attribute names → schema-agnostic functionality

• Core approach: Token Blocking
  1. Given a profile, extract all tokens that are contained in its attribute values
  2. Create one block for every distinct token with frequency > 2 → each block contains all profiles with the corresponding token

Pros:
  ▪ Parameter-free
  ▪ Efficient
  ▪ Unsupervised
Example of Token Blocking

**DATASET 1**

**Entity 1**
- name=United Nations Children’s Fund
- acronym=unicef
- headquarters=California

**Entity 2**
- name=Ann Veneman
- position=unicef
- address=California

**DATASET 2**

**Entity 3**
- organization=unicef
- hdq=California
- status=active

**Entity 4**
- firstName=Ann
- lastName=Veneman
- residence=California
• Block Building creates a huge number of blocks
• Results in an additional step in the workflow
• Goal: restructure the original blocks in order to increase **precision** at no significant cost in **recall**
• Focus on reducing / removing comparisons:
  1. **Redundant comparisons**, i.e., comparing profiles that were already compared in a previous block
  2. **Superfluous comparisons**, i.e., high number of comparisons between irrelevant profiles
• Collective approaches to tackle Variety
• Most methods crafted for Clean-Clean ER
• General outline:
  ◦ Iterative process starts with a few reliable seed matches
  ◦ Propagate initial matches to neighbors
  ◦ Order candidate matches in descending overall similarity
  ◦ Recompute the similarity of the neighbors
  ◦ Update candidate matches order
• Alternative is to perform a specific number of steps, rather than iterating until convergence
• Methods from previous generations are still applicable

• Only difference:
  • Similarity scores extracted in a schema-agnostic fashion, not from specific attributes
Veracity
+Volume
+Variety
+Velocity
Scope → Velocity & Variety & Volume & Veracity

• Applications with increasing data volume & time constraints
  ◦ Loose ones (e.g., minutes, hours) → Progressive ER
  ◦ Strict ones (i.e., seconds) → Real-time (On-line) ER

• End-to-end workflows for Progressive ER:

![Workflow Diagram]
Progressive Entity Resolution

- Unprecedented, increasing volume of data → applications can compromise with partial solutions to produce useful results

get most of the benefit much earlier

may require some pre-processing
Outline Progressive ER

- Requires:
  - Improved early quality
  - Same eventual quality

- Prioritization:
  - Defines **optimal processing order** for a set of entities
  -Static methods [1, 2]:
    - Guide which records to compare first *(independently* of ER matching results)*
  - Dynamic methods [3]:
    - If a duplicate is found, then check neighbors as well
    - Assumption: oracle for entity matching
Same workflow as original two workflows:

- Schema Alignment → Blocking → Matching → Clustering

Different goal:
- resolve each query over a large dataset in the shortest possible time (and with the minimum memory footprint)

Same scope (so far):
- structured data

Different input:
- stream of query profiles
Incremental Blocking

• Maintain a dynamic set of blocks
• I.e., block contents are updated as new profiles arrive
• Examples: DySimII [1], F-DySNI [2, 3], (S)BlockSketch [4]

Incremental Matching

• QDA [5] - SQL-like selection queries over a single dataset
• QuERy [6] - complex join queries over multiple, overlapping, dirty DSs
• Simple or statistical queries over possible worlds (i.e., resolutions) [7, 8]
• Evolving matching rules [9]

Incremental Clustering

• Maintain the entities detected by clustering
• Examples: Incremental Correlation Clustering [10]


• Crowdsourcing
  ◦ E.g.: how do maximize accuracy while minimize cost?
• Deep Learning
• Explainability for matches and non-matches
• Advance solutions for evolving data
  ◦ E.g., automatic configuration of workflows
• Algorithmic Bias
• Benchmarks
• …
Questions
– additional material & citations

ER Methods
ER Challenges

Veracity
• Structured data with known semantics and quality
• Dealing with high levels of profile noise

+ Volume
• Very large number of profiles

+ Variety
• Large volumes of semi-structured, unstructured or highly heterogeneous structured data

+ Velocity
• Increasing volume of available data
ER Challenges in time
Veracity
  + Volume
  + Variety
  + Velocity

ER Methods
• Earliest ER methods

• Scope:
  ◦ Structured data (e.g., small relational databases)

• Goal:
  ◦ Achieve high accuracy despite inconsistencies, noise, or errors in profiles

• Assumptions:
  ◦ Known schema → custom, schema-based solutions
Step 1: Schema Alignment

- **Scope:**
  - Record Linkage

- **Goal:**
  - Create mappings between equivalent attributes of the two schemata, e.g., *profession* $\equiv$ *job*

- **Types of Solutions:**
  - Instance-based: use profiles to learn mappings, transformations, rules (e.g., merge of two values)
  - Structure-based: use schema information by converting it to trees or graphs
  - Hybrid
## Step 1: Schema Alignment

- **Taxonomy of Main Schema Matching Methods**
  (in chronological order)

<table>
<thead>
<tr>
<th>Method</th>
<th>Category</th>
<th>Type of Evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cupid [1]</td>
<td>Structure-based</td>
<td>Name similarity, Constraints, Contextual similarity</td>
</tr>
<tr>
<td>Similarity Flooding [2]</td>
<td>Structure-based</td>
<td>Name similarity, Contextual similarity</td>
</tr>
<tr>
<td>COMA [3]</td>
<td>Hybrid</td>
<td>Name similarity, Constraints, Contextual similarity</td>
</tr>
<tr>
<td>Distribution-based [4]</td>
<td>Instance-based</td>
<td>Value distribution</td>
</tr>
</tbody>
</table>
• **Scope:**
  ◦ Both Deduplication and Record Linkage

• **Goal:**
  ◦ Entity resolution is an inherently quadratic problem, i.e., $O(n^2)$ with every profile compared to all others
  ◦ Blocking groups similar profiles into blocks
    ▪ Comparisons executed only inside each block
    ▪ Complexity is now quadratic to the size of the block (much smaller than dataset size!)
Computational cost

Input: Profile Collection \( P \)

\(|P| \) profiles

E.g.: For a dataset with 100,000 profiles:
~10\(^{10}\) comparisons,
If 0.05 msec each → >100 hours in total
1. Represent each profile by *one or more* signatures called **blocking keys**
   - Focus on **string values**

2. Place into blocks all profiles having the *same or similar* blocking key

3. Two matching profiles can be **detected** as long as they co-occur in at least one block
   - **Trade-off** between recall and precision!

<table>
<thead>
<tr>
<th>name</th>
<th>Ekaterini Ioannou</th>
</tr>
</thead>
<tbody>
<tr>
<td>zip code</td>
<td>9876</td>
</tr>
<tr>
<td>address</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Taxonomy of Blocking Methods [1]

<table>
<thead>
<tr>
<th>Method</th>
<th>Key Type</th>
<th>Matching awareness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suffix Arrays [3] + [4,5]</td>
<td>Hash-based</td>
<td>Static</td>
</tr>
<tr>
<td>MFIBlocks [7]</td>
<td>Hash-based</td>
<td>Static</td>
</tr>
<tr>
<td>Sorted Neighborhood [9] + [4,10]</td>
<td>Sort-based</td>
<td>Static</td>
</tr>
<tr>
<td>Sorted Blocks [12]</td>
<td>Hybrid</td>
<td>Static</td>
</tr>
<tr>
<td>ApproxDNF [13]</td>
<td>Hash-based</td>
<td>Static</td>
</tr>
<tr>
<td>Blocking Scheme Learner [14]</td>
<td>Hash-based</td>
<td>Static</td>
</tr>
<tr>
<td>CBlock [15]</td>
<td>Hash-based</td>
<td>Static</td>
</tr>
<tr>
<td>FisherDisjunctive [16]</td>
<td>Hash-based</td>
<td>Static</td>
</tr>
</tbody>
</table>
• Estimates the similarity of candidate matches

• Input: a set of blocks
  ◦ Every distinct comparison in any block is a candidate match

• Output: similarity Graph
  ◦ Nodes → profiles
  ◦ Edges → candidate matches
  ◦ Edge weights → matching likelihood (based on similarity score)
Evolution of Matching

Active Learning Methods [1,2]

Supervised Methods [3,4]

Probabilistic Methods [5,6]  Unsupervised Methods [7,8]

Learning-based Methods


All are heavily based on string similarity measures [6]
Step 4: Clustering

• Partitions the matched pairs into *equivalence clusters* i.e., groups of profiles describing the same real-world object

• Input
  ◦ *Similarity Graph*:
    ▪ Nodes → profiles
    ▪ Edges → candidate matches
    ▪ Edge weights → matching likelihood (based on similarity score)

• Output
  ◦ *Equivalence Clusters*
Relies on 1-1 constraint

- One profile from source dataset matches one profile from the target dataset

1. Unique Mapping Clustering \([1, 2]\)

- Sorts all edges in decreasing weight
- Starting from the top, each edge corresponds to a pair of duplicates, if:
  - None of the adjacent profiles have already been matched to other profiles, or
  - Predefined threshold < edge weight
Clustering Algorithms for Record Linkage

Relies on 1-1 constraint
  ◦ One profile from source dataset matches one profile from the target dataset

1. Unique Mapping Clustering
2. Row-Column Clustering [3]
  ◦ Based on an efficient approximation of the Hungarian Algorithm
Clustering Algorithms for Record Linkage

Relies on 1-1 constraint

- One profile from source dataset matches one profile from the target dataset

1. Unique Mapping Clustering
2. Row-Column Clustering
   - Efficient, heuristic solution to the assignment problem in unbalanced bipartite graphs
• A wealth of literature on clustering algorithms

• Requirements:
  ◦ Partitional and disjoint algorithms
    ▪ Sometimes overlapping may be desirable
  ◦ Goal: Sets of clusters that
    ▪ maximize the intra-cluster weights
    ▪ minimize the inter-cluster edge weights

Classification of clustering methods [6]

(slide from O. Hassanzadeh)
Dirty ER Clustering Algorithms Characteristics [3]

- Most important feature “unconstrained algorithms”, i.e.,
  - Algorithms need to be able to *predict* the correct number of clusters (do not require the number)
- Need to scale well
  - Time complexity < $O(n^2)$
- Need to be robust with respect to characteristics of the data
  - E.g., distribution of the duplicates
- Need to be capable of finding ‘singleton’ clusters
  - Different from many clustering algorithms
    - E.g., algorithms proposed for image segmentation

(slide from O. Hassanzadeh)
## Summary of Experimental Results [3]

<table>
<thead>
<tr>
<th>Method</th>
<th>Scalability</th>
<th>Ability to Find Correct Number of Clusters</th>
<th>Robustness Against</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Current Implementations)</td>
<td></td>
<td>Choice of Threshold</td>
</tr>
<tr>
<td>Partitioning</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>CENTER</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>MERGE CENTER</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Star</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>SR</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>BSR</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>CR</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>OCR</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Correlation Clustering</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Markov Clustering</strong></td>
<td>High</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Cut Clustering</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Articulation Point</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
</tbody>
</table>


Veracity
+Volume
+Variety
+Velocity
• Workflow remains the same
• Scope:
  ◦ (tens of) millions of structured profiles
• Goals:
  ◦ High accuracy despite noise
  ◦ High time efficiency despite the size of data
• Assumptions:
  ◦ Known schema → custom, schema-based solutions
Two types:

• Multi-core parallelization
  ◦ Single system → shared memory
  ◦ Distribute processing among available CPUs

• Massive parallelization
  ◦ Cluster of independent systems
  ◦ Map-Reduce paradigm [1]
    ▪ Data partitioned across the nodes of a cluster
    ▪ Map Phase: transforms a data partition into (key, value) pairs
    ▪ Reduce Phase: processes pairs with the same key
Mechanisms per Workflow step

• Blocking based on map reduce
  ◦ Dedoop [2]
  ◦ MapReduce-based Sorted Neighborhood [3]

• Matching
  ◦ Multi-core approaches [7, 8]
  ◦ MapReduce-based: Emphasis on load balancing
    ▪ BlockSplit & PairRange [4, 5]
    ▪ Dis-Dedup [6]
    ▪ Message-passing framework [9]

• Clustering
  ◦ Fast Multi-source ER (FAMER) framework [10, 11]
References

Veracity
+Volume
+Variety
+Velocity
Scope (e.g., user-generated Web Data):

- Voluminous, (semi-)structured datasets
  - **BTC09**: 1.15 billion triples, 182 million profiles
- Users are free to add attribute values and/or attribute names
  - Unprecedented levels of schema heterogeneity
    - **Google Base**: 100,000 schemata for 10,000 profile types
    - **BTC09**: 136,000 attribute names
- Several datasets produced by automatic information extraction techniques
  - Noise, tag-style values

Variety & Volume & Veracity

![Diagram of data processing steps: Schema Clustering, Block Building, Block Processing, Matching, Clustering]
Example of Web Data

DATASET 1

Entity 1
- name=United Nations Children’s Fund
- acronym=unicef
- headquarters=California
- address=Los Angeles, 91335

DATASET 2

Entity 3
- organization=unicef
- status=active
- address=Los Angeles, 91335

Entity 4
- firstName=Ann
- lastName=Veneman
- residence=California
- zip_code=90201

Loose Schema Binding
Split values
Attribute Heterogeneity
Noise
• Schema Matching → not applicable (too many alternatives)

• Instead, partition attributes according to their **syntactic** similarity, regardless of their **semantic** relation

• Goal: Facilitate next steps

• Scope: Both Clean-Clean and Dirty ER

• Attribute Clustering [1, 2, 3]
  - Create a graph, with nodes representing attributes
  - For each node $n_i$
    - Find the most similar node $n_j$
    - If $\text{sim}(n_i,n_j) > 0$, add an edge $<n_i,n_j>$
  ◦ Extract connected components
  ◦ Put all singleton nodes in a “glue” cluster
• Considers all attribute values and completely ignores all attribute names → schema-agnostic functionality

• Core approach: Token Blocking [1]
  1. Given a profile, extract all tokens that are contained in its attribute values
  2. Create one block for every distinct token with frequency $> 2$ → each block contains all profiles with the corresponding token

Pros:
  ▪ Parameter-free
  ▪ Efficient
  ▪ Unsupervised
Example of Token Blocking

**DATASET 1**

**Entity 1**
- name=United Nations Children’s Fund
- acronym=unicef
- headquarters=California

**Entity 2**
- name=Ann Veneman
- position=unicef
- address=California

**DATASET 2**

**Entity 3**
- organization=unicef
- hdq=California
- status=active

**Entity 4**
- firstName=Ann
- lastName=Veneman
- residence=California
Genealogy of Block Building Techniques [8]

- Token Blocking (TB) [1]
  - Attribute Clustering Blocking [2]
  - RDFKeyLearner [6]
  - Prefix-Infix(-Suffix) Blocking [3]
  - Semantic Graph Blocking [5]

MapReduce-based parallelizations in [7]
• Block Building resulted into huge number of blocks
• Results in addition of new step in the workflow
• Goal:
  o Restructure the original blocks in order to increase precision at no significant cost in recall
• Focus on reducing / removing comparisons:
  1. Redundant comparisons, i.e., comparing profiles that were already compared in a previous block
  2. Superfluous comparisons, i.e., high number of comparisons between irrelevant profiles
Block Processing Techniques

• Block Clustering:
  ◦ Operate at the level of entire blocks
  ◦ Methods
    ▪ Block Purging [1,2,3]
    ▪ Block Filtering [4]
    ▪ Block Clustering [5]

• Comparison Cleaning:
  ◦ Methods (next slide)
Comparison Cleaning Methods [17]

- Spectral Neighborhood (SPAN) [15]
- Comparison Propagation [6]
- Transitive LSH [16]
  - Comparison Pruning [7]
  - Weighted Edge Pruning (WEP) [8]

- Cardinality Edge Pruning (CEP) [8]
  - Extended Canopy Clustering [9,10]
  - Cardinality Node Pruning (CNP) [4,8]
  - Reciprocal Cardinality Node Pruning (ReCNP) [4]

- Weighted Node Pruning (WNP) [4,8]
  - Canopy Clustering [12]
  - BLAST [11]
  - Reciprocal Weighted Node Pruning (ReWNP) [4]

- Low Entity Co-occurrence Pruning (LECP) [13]
  - Low Block Co-occurrence Pruning (LBCP) [13]
  - Large Block Size Pruning (LBSP) [13]
  - CooSlicer [13]
  - Low Block Co-occurrence Excluder (LBCE) [13]
• Collective approaches to tackle Variety
• Most methods crafted for **Clean-Clean ER**
• **General outline of** SiGMa[1], PARIS[2], LINDA[3], RiMOM-IM[4,5]
  ◦ Iterative process starts with a few reliable seed matches
  ◦ Propagate initial matches to neighbors
  ◦ Order candidate matches in descending overall similarity
  ◦ Recompute the similarity of the neighbors
  ◦ Update candidate matches order
• MinoanER [6] performs a specific number of steps, rather than iterating until convergence
• Methods discussed before are still applicable
  ◦ Only difference: similarity scores extracted in a schema-agnostic fashion, not from specific attributes
• SplitMerge [1]
  ◦ Inherently capable of handling heterogeneous semantic types


4. Y. Ma, T. Tran. TYPiMatch: type-specific unsupervised learning of keys and key values for heterogeneous web data integration. WSDM 2013: 325-334


Veracity
+Volume
+Variety
+Velocity

ER Methods
Scope:

- Applications with increasing data volume & time constraints
  - Loose ones (e.g., minutes, hours) → Progressive ER
  - Strict ones (i.e., seconds) → Real-time (On-line) ER

- End-to-end workflows for Progressive ER:
Progressive Entity Resolution

• Unprecedented, increasing volume of data
  → applications can compromise with partial solutions to produce useful results

get most of the benefit much earlier

may require some pre-processing
• Requires:
  ◦ Improved early quality
  ◦ Same eventual quality

• Prioritization
  ◦ Defines optimal processing order for a set of entities
  ◦ Static methods [1, 2]:
    ▪ Guide which records to compare first (independently of ER matching results)
  ◦ Dynamic methods [3]:
    ▪ If a duplicate is found, then check neighbors as well
    ▪ Assumption: oracle for entity matching
Taxonomy of Static Prioritization Methods

- Sorted Neighborhood (SN)
- Standard Blocking (SAB)
- Token Blocking
- Meta-blocking (Blocking Graph)

- Progressive SN (PSN) [1]
- Hierarchy of Record Partitions (HRP) [1,4]
- Ordered List of Records (OLR) [1]
- Local SA-PSN (LS-PSN) [2]
- Global SA-PSN (GS-PSN) [2]
- Progressive Suffix Arrays Blocking (SA-PSAB) [2]
- Progressive Block Scheduling (PBS) [2]
- Progressive Profile Scheduling (PPS) [2]

- Schema-based
- Schema-agnostic
- Naive
- Advanced

- Comparison-based
- Block-based
- Profile-based
- Hybrid
Same workflow as original two workflows:

- Structured data

Same scope (so far):
- Structured data

Different input:
- stream of query profiles

Different goal:
- resolve each query over a large dataset in the shortest possible time (& with the minimum memory footprint)
Incremental Blocking
• DySimII [1] - extends Standard Blocking
• F-DySNI [2,3] - extends Sorted Neighborhood
• (S)BlockSketch [4] - bounded matching time, constant memory footprint

Incremental Matching
• QDA [5] - SQL-like selection queries over a single dataset
• QuERy [6] - complex join queries over multiple, overlapping, dirty DSs
• EAQP [7] - queries under data
• Evolving matching rules [8]

Incremental Clustering
• Incremental Correlation Clustering [9]


