Entity Resolution in the Web of Data

Part II

Kostas Stefanidis\textsuperscript{1}, Vasilis Efthymiou\textsuperscript{1,2}, Melanie Herschel\textsuperscript{3,4}, Vassilis Christophides\textsuperscript{5}

kstef@ics.forth.gr, vefthym@ics.forth.gr, melanie.herschel@lri.fr
vassilis christophides@technicolor.com

\textsuperscript{1}FORTH, \textsuperscript{2}University of Crete, \textsuperscript{3}Université Paris Sud, \textsuperscript{4}Inria Saclay, \textsuperscript{5}Paris R\&I Center, Tehcnicolor
From Part I

Entity resolution via blocking:
- Token blocking
- Attribute clustering
- Blocking based on infixes
Token Blocking vs Attribute Clustering

Matching pairs of entity descriptions

Attribute clustering uses a not so loose similarity function

Token blocking uses a loose similarity function

Set of all pairs of entity descriptions

e1 = {(name, Smith), (country, USA)}
e2 = {(about, R. Smith), (livesIn, California)}
e3 = {(brand, Jeep), (headquarters, USA)}
e4 = {(name, Ulrich), (country, Denmark)}
e5 = {(about, D. Brunson), (livesIn, Nevada)}
e6 = {(title, California Dreamin’), (length, 2:34)}
Prefix-Infix(-Suffix) - Evaluation

Matching pairs of entity descriptions

Infix Blocking

Infix Profile Blocking

Set of all pairs of entity descriptions
Entity Resolution in the Web of Data

So far…
Rely on the values of the descriptions
• A good way to handle data heterogeneity and low structuredness

=> Deal with loosely structured entities

=> Deal with various vocabularies (side effect)

Still, many redundant comparisons are performed!
• Can we also use the structural type of the descriptions?
For further enhancing efficiency of entity resolution

Block Post-Processing
Block Post-Processing

STEP 1
Block Building

STEP 2
Block Post-Processing
Block Post-Processing

The goal: Reduce further the number of comparisons

- **Remove oversized blocks**
  - Threshold on the number of descriptions in a block
- **Order blocks**
  - Examine first the blocks which are more likely to contain matches
- **Remove low-order blocks**
  - We do not gain much by examining them
- **Order comparisons**
  - Perform first the comparisons that are more likely to result in matches
- **Remove low-order comparisons**
  - Similar to removing low-order blocks
## Removing Oversized Blocks

<table>
<thead>
<tr>
<th></th>
<th>Eiffel</th>
<th>Tower</th>
<th>Liberty</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$e_1, e_2,$</td>
<td>$e_1, e_4,$</td>
<td>$e_2, e_3$</td>
</tr>
<tr>
<td></td>
<td>$e_3, e_4$</td>
<td>$e_5$</td>
<td></td>
</tr>
<tr>
<td>NY</td>
<td>$e_2, e_3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paris</td>
<td>$e_1, e_4$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1889</td>
<td>$e_1, e_4$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Block size threshold = 3
Removing Oversized Blocks

Block size threshold = 3
Block Post-processing

The goal: Reduce further the number of comparisons

- **Remove oversized blocks**
  - Threshold on the number of descriptions in a block

- **Order blocks**
  - Examine first the blocks which are more likely to contain matches

- **Remove low-order blocks**
  - We do not gain much by examining them

- **Order comparisons**
  - Perform first the comparisons that are more likely to result in matches

- **Remove low-order comparisons**
  - Similar to removing low-order blocks
Assign a **utility value** to each block:

- \( u(b_i) = \frac{\text{gain}(b_i)}{\text{cost}(b_i)} \)

\( \text{gain}(b_i) : \# \text{superfluous comparisons spared in subsequently examined blocks} \)

\( \text{cost}(b_i) : \# \text{comparisons entailed in } b_i \)

*Estimation for Clean-Clean Entity Resolution:* \( u(b_i) \approx \frac{1}{\max(|b_{i,1}|, |b_{i,2}|)} \)

\( b_{i,j} \) are the contents of block \( i \) that come from entity set \( j \)

Order the blocks in descending utility values

- This is the order in which they will be processed
- Low-order blocks will not be processed at all
Ordering Comparisons [Papadakis et al. 2011(b)] & [Whang et al. 2013]

Comparisons are ranked by the likelihood that they result in a match

E.g. based on the number of blocks they appear together [Papadakis et al. 2011b]

Match\_likelihood(e_i, e_j) = Jaccard(blocks(e_i), blocks(e_j)) =
|blocks(e_i) \cap blocks(e_j)| / |blocks(e_i) \cup blocks(e_j)|

Low-ordered comparisons are:
• performed last (irrespective of the block in which they appear) [Whang et al. 2013]
• not performed at all [Papadakis et al. 2011b]

This way, matches are identified faster!
Meta-Blocking

- **STEP 1**: Block Building
- **STEP 2**: Block Post-Processing
Meta-Blocking

**STEP 1**
Block Building

**STEP 2**
Meta-Blocking

**STEP 3**
Block Post-Processing
Meta-blocking [Papadakis et al. 2013 (b)]

A generic procedure for block reconstruction
- Create blocks resulting in fewer comparisons
- Preserve effectiveness

**Blocking graph**: abstract graph representation of the original set of blocks
- Nodes: entity descriptions
- Edges: connect descriptions co-occurring in blocks

Use the blocking graph for discarding **redundant comparisons**
- i.e. comparisons already performed

Prune edges, not satisfying a criterion, for discarding **superfluous comparisons**
- i.e. comparisons between non-matches
### Meta-blocking - Example

<table>
<thead>
<tr>
<th>name</th>
<th>Eiffel Tower</th>
<th>name</th>
<th>Statue of Liberty</th>
<th>name</th>
<th>Lady liberty</th>
</tr>
</thead>
<tbody>
<tr>
<td>architect</td>
<td>Sauvestre</td>
<td>architect</td>
<td>Bartholdi Eiffel</td>
<td>architect</td>
<td>Eiffel</td>
</tr>
<tr>
<td>year</td>
<td>1889</td>
<td>year</td>
<td>1886</td>
<td>location</td>
<td>NY</td>
</tr>
<tr>
<td>location</td>
<td>Paris</td>
<td>located</td>
<td>NY</td>
<td>location</td>
<td>Thessaloniki</td>
</tr>
</tbody>
</table>

**Blocks:**
(with token blocking)

- **Eiffel**
  - e₁, e₂, e₃, e₄
- **Tower**
  - e₁, e₄, e₅
- **Liberty**
  - e₂, e₃
- **NY**
  - e₂, e₃
- **Paris**
  - e₁, e₄
- **1889**
  - e₁, e₄

**13 comparisons**
to identify 2 matches

**Blocking graph:**

- edge weights = #common blocks

**Pruned blocking graph:**

- remove edges with weight < 2

- 2 comparisons
to identify 2 matches
Iterative blocking as a procedure of blocking post-processing
Iterative Blocking [Whang et al. 2009]

Entity resolution results of a processed block, may help identifying more matches in another block

- Newly created entity descriptions, i.e. merges of descriptions, are distributed to other blocks

Blocks are processed multiple times, until no new matches are found

Disk-based algorithm is used to scale the process

- Use segments, each fitting in the main-memory
Iterative Blocking - Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Architects</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td>Sauvestre</td>
<td>Paris</td>
</tr>
<tr>
<td>Statue of Liberty</td>
<td>1886</td>
<td>Bartholdi, Eiffel</td>
<td>NY</td>
</tr>
<tr>
<td>Lady Liberty</td>
<td>1885</td>
<td>Eiffel</td>
<td>Liberty Island, NY</td>
</tr>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td></td>
<td>Paris</td>
</tr>
<tr>
<td>Miss Liberty</td>
<td>1886</td>
<td>Gustave Eiffel</td>
<td>Liberty Island</td>
</tr>
</tbody>
</table>

Blocks generated if blocking keys are the year and the 1st letter of the location:
Iterative Blocking - Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Architects</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td>Sauvestre</td>
<td>Paris</td>
</tr>
<tr>
<td>Statue of Liberty</td>
<td>1886</td>
<td>Bartholdi, Eiffel</td>
<td>NY</td>
</tr>
<tr>
<td>Lady Liberty</td>
<td>1885</td>
<td>Eiffel</td>
<td>Liberty Island, NY</td>
</tr>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td></td>
<td>Paris</td>
</tr>
<tr>
<td>Miss Liberty</td>
<td>1886</td>
<td>Gustave Eiffel</td>
<td>Liberty Island</td>
</tr>
</tbody>
</table>

Blocks generated if blocking keys are the year and the 1st letter of the location:

1889
- e₁, e₄

1886
- e₂, e₅

1885
- e₃

P
- e₁, e₄

N
- e₂

L
- e₃, e₅

e₁, e₄ match! they are merged as e₁₄
## Iterative Blocking - Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Architects</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td>Sauvestre</td>
<td>Paris</td>
</tr>
<tr>
<td>Statue of Liberty</td>
<td>1886</td>
<td>Bartholdi, Eiffel</td>
<td>NY</td>
</tr>
<tr>
<td>Lady Liberty</td>
<td>1885</td>
<td>Eiffel</td>
<td>Liberty Island, NY</td>
</tr>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td></td>
<td>Paris</td>
</tr>
<tr>
<td>Miss Liberty</td>
<td>1886</td>
<td>Gustave Eiffel</td>
<td>Liberty Island</td>
</tr>
</tbody>
</table>

Blocks generated if blocking keys are the year and the 1st letter of the location:
Iterative Blocking - Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Architects</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td>Sauvestre</td>
<td>Paris</td>
</tr>
<tr>
<td>Statue of Liberty</td>
<td>1886</td>
<td>Bartholdi, Eiffel</td>
<td>NY</td>
</tr>
<tr>
<td>Lady Liberty</td>
<td>1885</td>
<td>Eiffel</td>
<td>Liberty Island, NY</td>
</tr>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td></td>
<td>Paris</td>
</tr>
<tr>
<td>Miss Liberty</td>
<td>1886</td>
<td>Gustave Eiffel</td>
<td>Liberty Island</td>
</tr>
</tbody>
</table>

Blocks generated if blocking keys are the year and the 1st letter of the location:

1889
\[ e_1, e_4, e_{14} \]

1886
\[ e_2, e_5 \]

1885
\[ e_3 \]

P
\[ e_1, e_4, e_{14} \]

N
\[ e_2 \]

L
\[ e_3, e_5 \]

e_2, e_5 match! they are merged as e_{25}
Iterative Blocking - Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Architects</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td>Sauvestre</td>
<td>Paris</td>
</tr>
<tr>
<td>Statue of Liberty</td>
<td>1886</td>
<td>Bartholdi, Eiffel</td>
<td>NY</td>
</tr>
<tr>
<td>Lady Liberty</td>
<td>1885</td>
<td>Eiffel</td>
<td>Liberty Island, NY</td>
</tr>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td></td>
<td>Paris</td>
</tr>
<tr>
<td>Miss Liberty</td>
<td>1886</td>
<td>Gustave Eiffel</td>
<td>Liberty Island</td>
</tr>
</tbody>
</table>

Blocks generated if blocking keys are the year and the 1st letter of the location:

- **1889**: e₁, e₄, e₁₄
- **1886**: e₂, e₅, e₂₅
- **1885**: e₃

- **P**: e₁, e₄, e₁₄
- **N**: e₂, e₂₅
- **L**: e₃, e₅, e₂₅

e₂, e₅ match! they are merged as e₂₅
Iterative Blocking - Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Architects</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td>Sauvestre</td>
<td>Paris</td>
</tr>
<tr>
<td>Statue of Liberty</td>
<td>1886</td>
<td>Bartholdi, Eiffel</td>
<td>NY</td>
</tr>
<tr>
<td>Lady Liberty</td>
<td>1885</td>
<td>Eiffel</td>
<td>Liberty Island, NY</td>
</tr>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td></td>
<td>Paris</td>
</tr>
<tr>
<td>Miss Liberty</td>
<td>1886</td>
<td>Gustave Eiffel</td>
<td>Liberty Island</td>
</tr>
</tbody>
</table>

Blocks generated if blocking keys are the year and the 1st letter of the location:
Iterative Blocking - Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Architects</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td>Sauvestre</td>
<td>Paris</td>
</tr>
<tr>
<td>Statue of Liberty</td>
<td>1886</td>
<td>Bartholdi, Eiffel</td>
<td>NY</td>
</tr>
<tr>
<td>Lady Liberty</td>
<td>1885</td>
<td>Eiffel</td>
<td>Liberty Island, NY</td>
</tr>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td></td>
<td>Paris</td>
</tr>
<tr>
<td>Miss Liberty</td>
<td>1886</td>
<td>Gustave Eiffel</td>
<td>Liberty Island</td>
</tr>
</tbody>
</table>

Blocks generated if blocking keys are the year and the 1st letter of the location:
## Iterative Blocking - Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Architects</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td>Sauvestre</td>
<td>Paris</td>
</tr>
<tr>
<td>Statue of Liberty</td>
<td>1886</td>
<td>Bartholdi, Eiffel</td>
<td>NY</td>
</tr>
<tr>
<td>Lady Liberty</td>
<td>1885</td>
<td>Eiffel</td>
<td>Liberty Island, NY</td>
</tr>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td></td>
<td>Paris</td>
</tr>
<tr>
<td>Miss Liberty</td>
<td>1886</td>
<td>Gustave Eiffel</td>
<td>Liberty Island</td>
</tr>
</tbody>
</table>

Blocks generated if blocking keys are the year and the 1\textsuperscript{st} letter of the location:
### Iterative Blocking - Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Architects</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td>Sauvestre</td>
<td>Paris</td>
</tr>
<tr>
<td>Statue of Liberty</td>
<td>1886</td>
<td>Bartholdi, Eiffel</td>
<td>NY</td>
</tr>
<tr>
<td>Lady Liberty</td>
<td>1885</td>
<td>Eiffel</td>
<td>Liberty Island, NY</td>
</tr>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td></td>
<td>Paris</td>
</tr>
<tr>
<td>Miss Liberty</td>
<td>1886</td>
<td>Gustave Eiffel</td>
<td>Liberty Island</td>
</tr>
</tbody>
</table>

Blocks generated if blocking keys are the year and the 1st letter of the location:

- 1889: $e_1, e_4, e_{14}$
- 1886: $e_2, e_5, e_{25}$
- 1885: $e_3$

$e_3, e_{25}$ match! they are merged as $e_{235}$
Iterative Blocking - Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Architects</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>e₁</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td>Sauvestre</td>
<td>Paris</td>
</tr>
<tr>
<td>e₂</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statue of Liberty</td>
<td>1886</td>
<td>Bartholdi, Eiffel</td>
<td>NY</td>
</tr>
<tr>
<td>e₃</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lady Liberty</td>
<td>1885</td>
<td>Eiffel</td>
<td>Liberty Island, NY</td>
</tr>
<tr>
<td>e₄</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td></td>
<td>Paris</td>
</tr>
<tr>
<td>e₅</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miss Liberty</td>
<td>1886</td>
<td>Gustave Eiffel</td>
<td>Liberty Island</td>
</tr>
</tbody>
</table>

Blocks generated if blocking keys are the year and the 1ˢᵗ letter of the location:

- e₃, e₂₅ match! they are merged as e₂₃₅
Iterative Blocking - Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Architects</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td>Sauvestre</td>
<td>Paris</td>
</tr>
<tr>
<td>Statue of Liberty</td>
<td>1886</td>
<td>Bartholdi, Eiffel</td>
<td>NY</td>
</tr>
<tr>
<td>Lady Liberty</td>
<td>1885</td>
<td>Eiffel</td>
<td>Liberty Island, NY</td>
</tr>
<tr>
<td>Eiffel Tower</td>
<td>1889</td>
<td></td>
<td>Paris</td>
</tr>
<tr>
<td>Miss Liberty</td>
<td>1886</td>
<td>Gustave Eiffel</td>
<td>Liberty Island</td>
</tr>
</tbody>
</table>

Blocks generated if blocking keys are the year and the 1st letter of the location:

process continues iteratively, until no new matches are found
Extend iterative blocking by using MinHash
HARRA [Kim & Lee 2010]

Extends iterative blocking by employing MinHash (for Jaccard approximation)

Scalability: A single hash table is used
- Before placing a description in a block, the description is compared to the contents of the block
**HARRA - Example**

e$_6$ should be placed in the blue bucket

**Hash Table:**

<table>
<thead>
<tr>
<th>Keys</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td>e$_1$, e$_2$</td>
</tr>
<tr>
<td>Red</td>
<td>e$_3$, e$_5$</td>
</tr>
<tr>
<td>Black</td>
<td>e$_4$</td>
</tr>
</tbody>
</table>
HARRA - Example

Before placing it there, we check if it matches \( e_1 \) or \( e_2 \)

\[
\begin{align*}
e_6 & = e_1 \quad \text{? NO} \\
e_6 & = e_2 \quad \text{? YES}
\end{align*}
\]

**Hash Table:**

<table>
<thead>
<tr>
<th>Keys</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td>( e_1 ) ( e_2 )</td>
</tr>
<tr>
<td>Red</td>
<td>( e_3 ) ( e_5 )</td>
</tr>
<tr>
<td>Black</td>
<td>( e_4 )</td>
</tr>
</tbody>
</table>
HARRA - Example

Before placing it there, we check if it matches $e_1$ or $e_2$

$e_6 = e_1$ ? NO

$e_6 = e_2$ ? YES

$e_{26}$ is the result of merging $e_6$ and $e_2$

$e_{26} = e_1$ ? NO

Hash Table:

<table>
<thead>
<tr>
<th>Keys</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td>$e_1$, $e_{26}$</td>
</tr>
<tr>
<td>Red</td>
<td>$e_3$, $e_5$</td>
</tr>
<tr>
<td>Black</td>
<td>$e_4$</td>
</tr>
</tbody>
</table>
HARRA - Example

Continue until:
• no merge occurs, OR
• saved comparisons > threshold, OR
• # iterations > constant

Re-initialize the input:  

<table>
<thead>
<tr>
<th>Keys</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td></td>
</tr>
<tr>
<td>Red</td>
<td></td>
</tr>
<tr>
<td>Black</td>
<td></td>
</tr>
</tbody>
</table>
Blocking vs Iterative Blocking

Matching pairs of entity descriptions

Blocking

Iterative Blocking

Set of all pairs of entity descriptions

Iterative Blocking (pros) Lead to more identified matches (cons) Lead to more comparisons
For handling huge volumes of data

MapReduce
MapReduce

Input data are partitioned

Input data partitions are sent to different nodes (mappers) in the cluster

• **Map phase**: distribute the current partition to multiple nodes (reducers)
  – Emit (key, value) pairs
  – Pairs with the same key are processed by the same reducer

• **Reduce phase**: process the pairs having the same key
  – Emit (key, value) pairs – the output of the program
MapReduce

For handling huge volumes of data:

Proceed entity resolution in partitions!

The map phase reflects blocking (re-distribute descriptions)

The reduce phase reflects entity resolution (check for matches)
## MapReduce – Input Data

<table>
<thead>
<tr>
<th>e1</th>
<th>e2</th>
<th>e3</th>
<th>e4</th>
<th>e5</th>
<th>e6</th>
<th>e7</th>
<th>e8</th>
</tr>
</thead>
</table>
MapReduce – Input Data Partitioning

\[ e_1 \quad e_2 \quad e_3 \quad e_4 \quad e_5 \quad e_6 \quad e_7 \quad e_8 \]
MapReduce – Mapper Input

Mapper 1
- e1
- e2
- e3

Mapper 2
- e4
- e5
- e6

Mapper 3
- e7
- e8

Mapper 4
- e8
MapReduce – Mapper Example

**Input:**
- e1 = {('name', 'Auguste Bartholdi'), ('year', 1834)}
- e2 = {('about', 'Auguste Bartholdi')}
- e3 = {('architects', 'Bartholdi Eiffel')}

**Output:**
- Bartholdi e1
- Auguste e1
- 1834 e1
- Bartholdi e2
- Auguste e2
- Eiffel e3
- Bartholdi e3
MapReduce – Mapper Output

Mapper 1

Mapper 2

Mapper 3
MapReduce – Shuffling & Sorting
MapReduce – Merging

Reducer 1

Reducer 2

Reducer 3

Reducer 4

Reducer 5
MapReduce – Reducer

Reducer 1

Reducer 2

Reducer 3

Reducer 4

Reducer 5
MapReduce – Reducer Example

Input:

| Bartholdi | e1 | e2 | e3 | e4 |

Output:

| e1-e2 | match |
| e3-e4 | match |
Dedoop – Standard Blocking \[\text{[Kolb et al. 2012]}\]

*Dedoop performs standard blocking using MapReduce*

**Map function**
- Input: an entity description
- Output: a (key, value) pair
  - key: the BKV of the description
  - value: the description having this BKV

The partitioning operates on the BKVs and distributes (key, value) pairs among reduce tasks
- All entities sharing the same BKV are assigned to the same reduce task

**Reduce function**: Computes in each block the similarities between all description pairs within the block
- Input: A BKV along with descriptions with this BKV
- Output: (key, value) pairs
  - key: a pair of descriptions
  - value: match/non-match
Dedoop – Mapper: BKVs as intermediate keys

e1 → k1, e1

k1

e2 → k2, e2

k2

k4

e3 → k4, e3

k4

e4 → k1, e4

k1

e5 → k3, e5

k3

k2

e6 → k2, e6

k2

e7 → k3, e7

k3

k4

e8 → k4, e8

k4
Dedoop – Mappers: Build Blocks
Dedoop – Reducers: Compare Block Contents

Reducer 1
- e1-e4 match

Reducer 2
- e2-e6 non-match
- e5-e7 non-match

Reducer 3
- e3-e8 match

Block 1:
- k1 e1
- k2 e2
- k4 e3
- k1 e4

Block 2:
- k2 e2
- k2 e6
- k3 e5

Block 3:
- k3 e5
- k3 e7

Block 4:
- k4 e3
- k4 e8
- k4 e8
Chaining MapReduce Jobs

The output of a MapReduce Job can be the input of another Job.

Chaining MapReduce reflects iterative entity resolution
Dedoop – Sorted Neighborhood [Kolb et al. 2011]

composite key = (partitionID, BKV)
partitionID(BKV) = 1, if BKV < “k3”
partitionID(BKV) = 2, else

(we know that we have two reducers available)
Dedoop SN: Sorting the Keys
Dedoop SN: Reducers Apply the Sliding Window
Dedoop SN: Reducers Apply the Sliding Window

Reducer 1

Window 1

e1-e4 match
e1-e2 match
e4-e2 non-match

Reducer 2

Window 1

e5-e7 match
e5-e3 non-match
e7-e3 match

w = 3
Dedoop SN: Reducers Apply the Sliding Window

Reducer 1

Window 2

- e4-e2 non-match
- e4-e6 match
- e2-e6 non-match

Reducer 2

Window 2

- e7-e3 match
- e7-e8 non-match
- e3-e8 match

w = 3
Dedoop SN: We Also Need To Compare The Boundary Entities

Reducer 1

Reducer 2

1.k1 e1
1.k2 e2
2.k4 e3
1.k1 e4
1.k2 e2
1.k2 e6
2.k3 e5
1.k2 e6
2.k3 e7
2.k4 e3
2.k4 e8

w = 3

e2-e5 match?

.....
e6-e7 match?
Dedoop SN: Reducers Also Output the Boundary Descriptions

Add a **boundary number prefix** to the output composite keys

Reducer 1

```
1.k1 e1
1.k1 e4
1.k2 e2
1.k2 e6
2.k3 e5
2.k3 e7
```

Reducer 2

```
2.k3 e3
2.k4 e8
```

**Boundary number:**
The last \(w-1\) descriptions of reducer \(i\) are assigned the boundary number \(i\).

The first \(w-1\) descriptions of reducer \(i+1\) are also assigned the boundary number \(i\).

The actual blocking key of \(e5\) is \(k3\), it was assigned to reducer 2 and it is associated with boundary number 1.

**w = 3**
Dedoop SN: New MapReduce Job for the Boundary Pairs
Dedoop SN: Partition by Boundary Number

Reducer 1

Reducer 2

Reducer applies sliding window

Window 1

Window 2

Identical map

e2-e6 non-match

e2-e5 match

e6-e5 non-match

e6-e7 match

e5-e7 match

e6-e5 non-match
Still, there are repeated comparisons
Dedoop SN: Skipping Repeated Comparisons

These comparisons are not performed again: They have been performed in the previous MapReduce job (they come from the same reducer)
Don’t match twice [Kolb et al. 2013]

Overlapping blocks lead to repeated comparisons

Adopt Comparison Propagation [Papadakis et al. 2012] to MapReduce:

• Descriptions need to be compared only within their least common block
Overlapping Blocks Lead to Repeated Comparisons
Map: Append the Subset of Smaller Keys for the Same Description

<table>
<thead>
<tr>
<th>e1</th>
<th>e2</th>
<th>e3</th>
<th>e4</th>
<th>e5</th>
<th>e6</th>
<th>e7</th>
<th>e8</th>
</tr>
</thead>
<tbody>
<tr>
<td>k1 e1, {}</td>
<td>k2 e1, {k1}</td>
<td>k3 e1, {k1, k2}</td>
<td>k1 e2, {}</td>
<td>k2 e2, {k1}</td>
<td>k1 e3, {}</td>
<td>k4 e3, {k1}</td>
<td>k1 e4, {}</td>
</tr>
<tr>
<td>k1 e5, {}</td>
<td>k2 e5, {k1}</td>
<td>k3 e5, {}</td>
<td>k5 e5, {k3}</td>
<td>k2 e6, {}</td>
<td>k4 e6, {k2}</td>
<td>k5 e6, {k2, k4}</td>
<td>k3 e7, {}</td>
</tr>
<tr>
<td>k4 e8, {}</td>
<td>k5 e8, {k4}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Map: Append the Subset of Smaller Keys for the Same Description
Resulting Comparisons

- e1-e2
- e1-e3
- e1-e4
- e2-e3
- e2-e4
- e3-e4
- e1-e5
- e1-e6
- e1-e7
- e2-e6
- e4-e6
- e3-e6
- e3-e8
- e6-e8
Assume that there is a rule $R: \text{Match}(e_1, e_2) \Rightarrow \text{Match}(e_4, e_5)$ and that we have inferred: $\text{Match}(e_1, e_2)$

In C2, we cannot infer $\text{Match}(e_4, e_5)$

We should somehow inform C2 that $e_1$ matches $e_2$

- Then we could infer that $e_4$ matches $e_5$, according to rule $R$

Solution: message passing

- After matching in C1 finishes, send a message “Match($e_1$, $e_2$)”
- In the next MapReduce round, entity resolution runs with the new evidence and infers $\text{Match}(e_4, e_5)$
Linda [Böhm et al. 2012]

- Works on an entity graph constructed from RDF triples having URIs as subject, predicate and object
  - Literals are stored for each entity \( e \) as \( L(e) \)

- Matches are identified using two kinds of similarities:
  - String similarity (token-based) of their literal values \( L(e) \)
    - Checked once
  - Contextual similarity (based on neighbors in the entity graph)
    - Checked iteratively
What is context?

- Let node $n$ in an entity graph correspond to an RDF subject or object, identified by a URI.
- The context $C(n)$ of $n$ is a set of tuples $(p_i, z_i, w_i)$, where
  - $z_i$ is a neighboring node of $n$.
  - $p_i$ is the predicate associated with an edge connecting $n$ with $z_i$.
  - $w_i$ is a numeric weight (how discriminative this information is).

That is, the context of $n$ includes objects $z_i$ of triples with $n$ as subject and subjects $z_i$ of triples with $n$ as object.

$$C(\text{Statue of liberty}) = \{(\text{location, Liberty Island, w1}), (\text{is work of, Bartholdi, w2})\}$$
Contextual Similarity

The contextual similarity of nodes n and m is:

\[ \text{context}_\text{sim}(n, m) = \]

\[ \sum_{(p_i, z_i, w_i) \in C(n)} \max_{(p_j, z_j, w_j) \in C(m)} w_i \cdot x_{z_i, z_j} \cdot \text{sim}(p_i, p_j), \text{if } |C(n)| \leq |C(m)| \]

\[ \sum_{(p_j, z_j, w_j) \in C(m)} \max_{(p_i, z_i, w_i) \in C(n)} w_j \cdot x_{z_i, z_j} \cdot \text{sim}(p_i, p_j), \text{else} \]

where

- \( x_{n,m} \) is 1, if n, m are identified as matches, and 0, else
- \( \text{sim}(p_i, p_j) \) is the string similarity of the predicates of n, m

*Intuitively, the contextual similarity finds matching neighbors and sums up their similarity values*
Contextual Similarity

Overall similarity: combine sim and context_sim

The similarity score for descriptions n and m is:

$$\text{sim}(n, m) + \beta \cdot \text{context\_sim}(C(n), C(m)) - \theta$$

- $\beta$ controls the contextual influence
- $\theta$ is used for re-normalization to values around 0
- positive scores reflect likely mappings
- negative scores imply dissimilarities

Experiments have shown $\beta = 1$ to perform well
**Linda** [Böhm et al. 2012]

**Scalability:** Entity graph partitions are processed in parallel
- Each MapReduce node holds:
  - A partition of the graph along with the similarities of the entity description pairs in this partition
- Entity pairs are stored in a priority queue in descending order wrt. their similarity

**Effectiveness:** Messages from mappers to reducers, only for the entity pairs that need similarity re-computation
LINDA Algorithm

Two square matrices (|E|x|E|) are used:

- X captures the **identified matches** (binary values)
- Y captures the **pair-wise similarities** (real values)
  - Initialization: common neighbors and string similarity of literals
  - Updates: Use the new identified matches of X

Until the priority queue (extracted from Y) becomes empty:

- Get the pair (e_i, e_j) with the highest similarity
  - (e_i, e_j) match by default!
  - Update X: matches of e_i are also matches of e_j
- Update the queue wrt. the new matches
Distribute across a cluster the input entity graph
- A node $i$ holds a portion $Q_i$ of the priority queue and the respective part $G_i$ of the graph

**Map phase**
- Mapper $i$ reads $Q_i$ and forwards messages to reducers for similarities recomputations
  - Matrix $X$ of identified matches is updated

**Reduce phase**
- Similarities re-computations (Matrix $Y$)
- Updates on priority queues
Priority Queue:

- (dbpedia:Statue_of_Liberty, yago:Statue_of_Liberty)
- (dbpedia:Statue_of_Liberty, yago:Liberty_Island)
- (dbpedia:Liberty_Island, yago:Upper_NY_Bay)
- (dbpedia:Liberty_Island, yago:Liberty_Island)
- (dbpedia:Liberty_Island, yago:Statue_of_Liberty)
- (dbpedia:Bartholdi, fb:m.0jph6)
- (dbpedia:Bartholdi, yago:Statue_of_Liberty)
- (dbpedia:Bartholdi, fb:m.072p8)
Priority Queue 1 (machine 1):

- (dbpedia:Statue_of_Liberty, yago:Statue_of_Liberty)
- (dbpedia:Statue_of_Liberty, yago:Liberty_Island)
- (dbpedia:Liberty_Island, yago:Upper_NY_Bay)
- (dbpedia:Liberty_Island, yago:Liberty_Island)
- (dbpedia:Liberty_Island, yago:Statue_of_Liberty)

Priority Queue 2 (machine 2):

- (dbpedia:Bartholdi, fb:m.0jph6)
- (dbpedia:Bartholdi, yago:Statue_of_Liberty)
- (dbpedia:Bartholdi, fb:m.072p8)

The priority queue is partitioned and partitions are sent to the MapReduce nodes.
The priority queue is partitioned and partitions are sent to the MapReduce nodes.
Priority Queue 1:

| (dbpedia:Statue_of_Liberty, yago:Statue_of_Liberty) |
| (dbpedia:Statue_of_Liberty, yago:Liberty_Island) |
| (dbpedia:Liberty_Island, yago:Upper_NY_Bay) |
| (dbpedia:Liberty_Island, yago:Liberty_Island) |
| (dbpedia:Liberty_Island, yago:Statue_of_Liberty) |

Priority Queue 2:

| (dbpedia:Bartholdi, fb:m.0jph6) |
| (dbpedia:Bartholdi, yago:Statue_of_Liberty) |
| (dbpedia:Bartholdi, fb:m.072p8) |

The head of each queue is a match by default
This triggers update messages
Dequeue these pairs, as each entity can be mapped to at most one entity per data source.
Priority Queue 1:

(dbpedia:Statue_of_Liberty, yago:Statue_of_Liberty)

(dbpedia:Statue_of_Liberty, yago:Liberty_Island)

(dbpedia:Liberty_Island, yago:Upper_NY_Bay)

(dbpedia:Liberty_Island, yago:Liberty_Island)

(dbpedia:Liberty_Island, yago:Statue_of_Liberty)

Priority Queue 2:

(dbpedia:Bartholdi, fb:m.0jph6)

(dbpedia:Bartholdi, yago:Statue_of_Liberty)

(dbpedia:Bartholdi, fb:m.072p8)

Send messages to the other nodes and check this constraint again
Priority Queue 1:

(dbpedia:Liberty_Island, yago:Upper_NY_Bay)
(dbpedia:Liberty_Island, yago:Liberty_Island)

Priority Queue 2:

Contextual similarity re-computations
Property names are also taken into account
Priority Queue 1:

- (dbpedia:Liberty_Island, yago:Liberty_Island)
- (dbpedia:Liberty_Island, yago:Upper_NY_Bay)

Priority Queue 2:

Priority queues are updated based on the new similarities
Priority Queue 1:

(dbpedia:Liberty_Island, yago:Liberty_Island)

(dbpedia:Liberty_Island, yago:Upper_NY_Bay)

Priority Queue 2:

The head of each queue is a match by default
This triggers update messages
Dequeue this pair, as each entity can be mapped to at most one entity per data source
Priority Queue 1:

Priority Queue 2:

Output mappings
Using Neighbors for Computing Similarities

Matching pairs of entity descriptions

Without neighbors (a loose similarity function is used)

With neighbors (a strict similarity function is used)

Set of all pairs of entity descriptions

With neighbors (pros) Lead to more identified matches
(cons) Lead to more comparisons
Entity Resolution in the Web of Data

So far…
Rely on the values and relations of the descriptions
• *A good way to handle data heterogeneity and low structuredness*

=> Deal with loosely structured entities
=> Deal with various vocabularies *(side effect)*
=> Deal with large volumes of data

Still, many redundant comparisons are performed!
• Can we also use the structural type of the descriptions?
Tutorial Overview

What follows in Part II:

• **Objectives of methods**
  – Effectiveness
  – Efficiency
  – Scalability

• **Learning for Entity Resolution** [just the general picture]

• **Conclusions** (~20 mins)
Objectives of Entity Resolution Methods

- **Effectiveness**
  - Maximize the number of true matches
  - Minimize the number of false matches and false non-matches
- **Efficiency**
  - Minimize the number of performed comparisons
- **Scalability** (for handling large volumes of data)
  - Distribute the task of entity resolution to multiple computational resources, e.g. MapReduce

*The difference between efficiency and scalability*
- An efficient method could be limited to a specific data size
- A scalable method could work in a distributed approach, without skipping any redundant comparisons
Effectiveness

Effectiveness, typically, by iterating over the data until no new matches are found.

To measure effectiveness:
- A ground truth is required, i.e. a correct result of entity resolution for a given set of descriptions.

Effectiveness is measured by:
- Precision
- Recall
- F-score
• **Precision**: number of correctly identified matches, compared to the number of all suggested matches (correctly or incorrectly)

\[
\text{Precision} = \frac{\text{\#identified \_true\_matches}}{\text{\#suggested\_matches}}
\]

• **Recall**: number of correctly identified matches, compared to the actual number of matches

\[
\text{Recall} = \frac{\text{\#identified \_true\_matches}}{\text{\#true\_matches}}
\]

• **F-score** (or F-measure): the harmonic mean of precision and recall

\[
\text{F-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\]
Measures for Effectiveness

**Generalized merge distance (GMD)** [Menestrina et al. 2010]  
*inspired by edit distance*

- **GMD(X, Y):** The minimum cost of transforming the result X of an entity resolution method to the ground truth Y
  - For transformation use two set operations, **split** and **merge**
  - The cost for transforming X to Y is the sum of the costs of the splits and merges needed
Let the cost of splitting be 2 and the cost of merging be 1:

**Ground truth** $Y: \{ (e_1, e_2), (e_3, e_4) \}$

**Entity Resolution Output** $X: \{ (e_1), (e_2), (e_3, e_4) \}$

**Transformation (merge):** $(e_1), (e_2) \rightarrow (e_1, e_2)$

Cost: 1

**GMD** $(X, Y) = 1$
Let the cost of splitting be 2 and the cost of merging be 1:

**Ground truth Y:** \{(e_1, e_2), (e_3, e_4)\}

**Entity Resolution Output X':** \{(e_1, e_2, e_3), (e_4)\}

**Transformation (split):** \{(e_1, e_2, e_3), (e_4)\} \rightarrow \{(e_1, e_2), (e_3), (e_4)\}  \text{ Cost: 2}

**Transformation (merge):** \{(e_1, e_2), (e_3), (e_4)\} \rightarrow \{(e_1, e_2), (e_3, e_4)\}  \text{ Cost: 1}

\[
\text{GMD}(X', Y) = 3
\]
Measures for Effectiveness

**Evaluate also the intermediate results of blocking, i.e. a blocking collection**

- Pairs of descriptions in the same block denote candidate matches
- **Pairs quality** corresponds to precision
- **Pairs completeness** corresponds to recall
Measures for Effectiveness

Evaluate also the intermediate results of blocking, i.e. a blocking collection
- Pairs of descriptions in the same block denote candidate matches
- **Pairs quality** corresponds to precision
- **Pairs completeness** corresponds to recall

**Blocking cardinality** (BC) approximates pairs completeness
- BC defines the average num of blocks an entity description is placed in

\[
BC = \frac{\sum_{b_i \in B} |b_i|}{|E|}
\]

- \(b_i\): a block in a blocking collection B
- \(E\): a given set of descriptions

\(BC\) reflects the degree of overlap of a blocking collection
- In partitioning blocks, \(BC = 1\)
- In overlapping blocks, \(BC > 1\)
Objectives of Entity Resolution Methods

- Effectiveness
- **Efficiency**
  - Minimize the number of performed comparisons
- Scalability
Efficiency

Comparisons between entity descriptions are \textit{computationally expensive operations in the process of entity resolution}

The goal is to:
Minimize the number of comparisons

How?
– Use blocking
– Use other block post-processing methods
  • \textit{i.e. methods for processing the generated blocks to reduce further the number of comparisons}
**Measures for Efficiency**

**Reduction ratio (RR):** A metric for efficiency in the context of blocking

Assume a blocking collection $B$:

*RR measures the ratio of comparisons that will not be performed when using $B$ over the number of comparisons required by a different collection $B'$ that either includes blocking, or not*

$$RR = 1 - \frac{|C_B|}{|C_{B'}|}$$

$|C_B|$ is the total number of comparisons contained in $B$:

$$|C_B| = \sum_{b_i \in B} \frac{|b_i| \cdot (|b_i| - 1)}{2}$$

*assuming symmetry holds*

**E.g. if** $B = \{(e_1, e_2), (e_1, e_3, e_4)\}$, then

$C_B = \{(e_1, e_2), (e_1, e_3), (e_1, e_4), (e_3, e_4)\}$, and $|C_B| = 4$

**match**(e$_1$, e$_2$) $\Rightarrow$ match (e$_2$, e$_1$)
Measures for Efficiency

Comparison cardinality (CC) approximates the reduction ratio

- CC is the average number of block assignments per comparison

\[ CC = \frac{\sum_{b_i \in B} |b_i|}{|C_B|} \]

[Papadakis et al. 2012]

In general, CC reflects the distribution of comparisons per block
Measures for Efficiency

BC and CC form two orthogonal axes of a metric space, capturing the tradeoff between effectiveness and efficiency.
Objectives of Entity Resolution Methods

- Effectiveness
- Efficiency
- **Scalability** (for handling large volumes of data)
  - Distribute the task of entity resolution to multiple computational resources, e.g. MapReduce
Scalability

*Scalable methods can handle entity resolution in large volumes of data, namely in the scale of millions or billions of entity descriptions*

Usually, such methods use a distributed approach

- Parallelize the process of entity resolution across multiple computational resources

**A common way of measuring scalability**

Plot the ratio of runtime needed by an entity resolution method to the size of the input data
Measures for Scalability

**Speedup** $S_p$: how much a parallel algorithm that uses $p$ processors is faster than a corresponding sequential algorithm

\[
S_p = \frac{T_1(\text{sequential})}{T_p(\text{parallel})}
\]

$T_1$: the execution time of the sequential algorithm and $T_p$: the execution time of the parallel algorithm, using $p$ processors

*The ideal speedup is linear, i.e. doubling the number of processors halves the execution time*
Tutorial Overview

What follows in Part II:

• **Learning for Entity Resolution** [just the general picture]

• Conclusions
Learning for Entity Resolution

Entity resolution in other words…

Given a vector of attribute-wise similarities for a pair of entity descriptions \((e_i, e_j)\), compute the probability \(P(e_i \text{ and } e_j \text{ match})\)

Take a decision on this problem!

[Elmagarmid et al. 2007, Getoor & Machanavajjhala 2012]

What is a vector of attribute-wise similarities, or comparison vector?

– Keep the result of comparing the values of a pair \((e_i, e_j)\) of descriptions
  E.g. \(x_{e_i,e_j} = [0.3, 0.7, 0.2]\)

This problem definition implies entity descriptions with the same set of attributes, i.e. data with high structuredness
Learning for Entity Resolution

Is it easy to compute $P(e_i \text{ and } e_j \text{ match})$?

*Learning helps towards automating this task*

- Given a set of descriptions $E$, take a decision on matches/non-matches, based on the following rule

$$R = \frac{P(\gamma | q \in M)}{P(\gamma | q \in Q)}$$

$q = (e_i, e_j)$, $\gamma$ is the comparison vector of $e_i, e_j$

$M, Q$ is the matching, non-matching pairs of descriptions in $E$

[Fellegi & Sunter 1969]
Learning for Entity Resolution

The decision of a match/non-match is based on a threshold $t$

If $R$ is greater than a threshold value $t$, $q$ is a match

Otherwise, it is a non-match

$R > t \Rightarrow q \in M$

$R \leq t \Rightarrow q \in Q$

Extension [Fellegi & Sunter 1969]

Use a third set $A$ for ambiguous pairs of descriptions, i.e. neither matches nor non-matches ($t' < t$)

$R > t \Rightarrow q \in M$

$t' \leq R \leq t \Rightarrow q \in A$

$R < t' \Rightarrow q \in Q$

In brief, existing approaches use:

- Supervised learning techniques, active learning techniques, unsupervised learning techniques
Conclusions
Entity resolution is the problem of identifying descriptions of the same entity within one or across multiple data sources.
Different types of input data impose different solutions for the problem of entity resolution.
Effectiveness: Find as many (few) true (false) matches as possible

Efficiency: Resolve the given descriptions as fast as possible, e.g. by reducing redundant comparisons
  - Pre-processing to place descriptions in blocks

Scalability: Methods that can cope with Big Data
  - Distribute the task of entity resolution to multiple computational resources, e.g. Map/Reduce

Discern entity resolution methods wrt. their main objective
Solution Space

**Type of method input**

- **Graph**
- **Tree**
- **Tabular**

**Objective of method**

- **Effectiveness**
- **Efficiency**
- **Scalability**

**Type of method**

- **Blocking**: Group together descriptions close to each other
  - Rely on blocking keys, i.e. criteria for placing descriptions into blocks
- **Iterative**: Identify matches that can lead to new matches
  - E.g. use the already merged descriptions
- **Learning**: Use training data, annotated as matches or not
  - Classify descriptions, using statistical inference
Solution Space – A Detailed Taxonomy

Approaches for entity resolution

Blocking approaches
- Partitioning
- Overlapping
  - Overlap-positive
  - Overlap-negative
  - Overlap-neutral

Iterative approaches
- Matching-based
- Merging-based

Learning approaches
- Supervised
- Active
- Unsupervised
Partitioning vs. Overlapping Blocks

Blocking approaches are distinguished between:

• **Partitioning**: Each description is placed in exactly one block
  – Fewer comparisons

• **Overlapping**: Each description can be placed in more than one block
  – More identified matches

In overlapping approaches, *the number of common blocks between two descriptions can be an indication of their similarity*

• **Overlap-positive**: many common blocks $\rightarrow$ very similar
• **Overlap-negative**: few common blocks $\rightarrow$ very similar
• **Overlap-neutral**: #common blocks is irrelevant
Discussion on Blocking

Blocking increases the speed of entity resolution
- Cost: missed matches

Selecting a good blocking key is more important than the blocking technique [Christen 2012]

Partitioning approaches save space and time
- Fewer, smaller blocks, resulting in less comparisons

Overlapping approaches return more matches
- Trade-off between the number and the size of the blocks:
  - Few, large blocks vs. many, small blocks
    - More comparisons vs. more missed matches
  Overlap-positive: lower misclassification cost
    - Seem more appropriate for the Web of data
## A Classification of Blocking Approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Partitioning</th>
<th>Overlapping</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>positive</td>
</tr>
<tr>
<td>Fellegi &amp; Sunter 1969</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Hernandez &amp; Stolfo 1995</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yan et al. 2007</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Draisbach &amp; Naumann 2009</td>
<td></td>
<td></td>
</tr>
<tr>
<td>McCallum et al. 2000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Christen 2012</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gravano et al. 2001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aizawa &amp; Oyama 2005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jin et al. 2003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kolb et al. 2011, 2012</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Papadakis et al. 2011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Papadakis et al. 2013 (a)</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Papadakis et al. 2013 (b)</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Papadakis et al. 2012</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

*: tabular data  
+: graph data
Iterative Approaches

Partial results of the entity resolution process can be propagated to generate new results

Iterative approaches can be grouped into:

– **Matching-based**: Exploit relationships between entity descriptions
  
  • *If descriptions related to* $e_i$ *are similar to descriptions related to* $e_j$, *this is an evidence that* $e_i$ *and* $e_j$ *are also similar*

– **Merging-based**: Exploit the partial results of merging descriptions
Discussion on Iterative Approaches

Iterative approaches target high *effectiveness*
- Exhaustively consider candidate matches

Each iteration is based on new knowledge
- Identified matches
- Merged descriptions of identified matches

Hybrid methods, i.e. *iterative blocking*, benefit from:
- The *efficiency* of blocking approaches
- The *effectiveness* of iterative approaches

Iterative approaches seem to fit well to graph data
- Relationships between descriptions are an important part of the available semantics
A Classification of Iterative Approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Matching-based</th>
<th>Merging-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bhattacharya &amp; Getoor 2004, 2007</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Rastogi et al. 2011</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Dong et al. 2005</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Herschel et al. 2012</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Weis &amp; Naumann 2006</td>
<td>□</td>
<td></td>
</tr>
<tr>
<td>Weis &amp; Naumann 2004</td>
<td>□</td>
<td></td>
</tr>
<tr>
<td>Leitão et al. 2007, 2013</td>
<td>□</td>
<td></td>
</tr>
<tr>
<td>Puhlmann et al. 2006</td>
<td>□</td>
<td></td>
</tr>
<tr>
<td>Böhm et al. 2012</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Benjelloun et al. 2009</td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>Benjelloun et al. 2007</td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>Whang et al. 2009</td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>Kim &amp; Lee 2010</td>
<td></td>
<td>•</td>
</tr>
</tbody>
</table>

• : tabular data
□ : tree data
+ : graph data
Discussion

**Type of method input:**
- Determines the complexity of the similarity measure

**Objective of method:**
- Effectiveness is achieved by increasing the number of comparisons in a single or multiple iterations
  - **Iterative** methods target high effectiveness

- Efficiency is achieved by reducing the number of comparisons
  - **Blocking** methods target high efficiency

- Scalable methods are capable of exploiting multiple machines
  - Similarity computation should be parallelizable
A Classification of Entity Resolution Approaches

Next, a classification on entity resolution approaches wrt. the type of their input data, the type of their method and their objectives

- ☐ indicates focus on efficiency
- • indicates focus on effectiveness
- + indicates focus on scalability
<table>
<thead>
<tr>
<th>Type of method input</th>
<th>Approach</th>
<th>Blocking</th>
<th>Iterative</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tabular</td>
<td>Fellegi &amp; Sunter 1969</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hernandez &amp; Stolfo 1995</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yan et al. 2007</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Draisbach &amp; Naumann 2009</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>McCallum et al. 2000</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Christen 2012</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gravano et al. 2001</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Aizawa &amp; Oyama 2005</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Jin et al. 2003</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kolb et al. 2011, 2012</td>
<td>☐ +</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Benjelloun et al. 2009</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Benjelloun et al. 2007</td>
<td>☐</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Whang et al. 2009</td>
<td>☐ ☐ +</td>
<td>☐ ☐ +</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kim &amp; Lee 2010</td>
<td>☐ ☐ +</td>
<td>☐ ☐ +</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Herschel et al. 2012</td>
<td>☐ +</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dong et al. 2005</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bhattacharya &amp; Getoor 2004, 2007</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rastogi et al. 2011</td>
<td>☐</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Cochinwala et al. 2001</td>
<td>☐</td>
<td></td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>Bilenko &amp; Mooney 2003</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Christen 2008</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Chen et al. 2009</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ravikumar &amp; Cohen 2004</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bhattacharya &amp; Getoor 2006</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sarawagi &amp; Bhamidipaty 2002</td>
<td>☐</td>
<td></td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>Tejada et al. 2002, 2001</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wang et al. 2012</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verykios et al. 2000</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Weis &amp; Naumann 2006</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Weis &amp; Naumann 2004</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Leitao et al. 2007, 2013</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Puhlmann et al. 2006</td>
<td>☐</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Papadakis et al. 2011</td>
<td>☐</td>
<td></td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>Papadakis et al. 2013 (a)</td>
<td>☐</td>
<td></td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>Papadakis et al. 2013 (b)</td>
<td>☐ ☐ +</td>
<td>☐ ☐ +</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Papadakis et al. 2012</td>
<td>☐</td>
<td></td>
<td>☐ +</td>
</tr>
<tr>
<td></td>
<td>Bohm et al. 2012</td>
<td>☐</td>
<td></td>
<td>+</td>
</tr>
</tbody>
</table>
Open Issues
Open Issues

Similarity measures

- Measures need to consider structural, value and contextual similarities between entities
  - Take into account *low structuredness, incompleteness, erroneous values, various vocabularies, different formats* of Web data

Inter-relationships between entity descriptions

- A traditional focus: Discover equality links between descriptions
  - sameAs links
- To improve data interlinking, infer other relationships
  - located in, related to, part of links
- From a different point of view: When such relationships are available, use them for enhancing the matching process
Open Issues

**Large-scale entity resolution using MapReduce**
- Few approaches/adaptations appeared only recently
  - We can do more for effectiveness!

**Temporal entity resolution**
- Entity resolution should account for changes over time
  - The Web evolves constantly with large volumes of new data and updates
    - E.g. an update in the family status of a person, should not result in not matching an updated description of this person with another description not updated
- Yago2 [Hoffart et al. 2012]: A temporal knowledge base, built with data from Wikipedia, GeoNames and Wordnet
Open Issues

Probabilistic entity resolution
• *The results of entity resolution sometimes are not accurate*
  – Due to data heterogeneity, the evolving nature of data, etc.
• A possible solution: Associate the identified matches with a belief score
  – Scores can be based on the quality of the source, e.g. wrt. outdated or erroneous data

Querying for entities
• Entity resolution at query time: *Ask for entities relevant to a specific query*
  – Two stages of processing:
    • Extract the relevant entity descriptions
    • Resolve the extracted entities
• *Interestingly, query time entity resolution enables an exploratory search among entities*
Thank You!

Other points for future work?

Questions?
References
References

References

References

References

References

- Kolb, L., Thor, A., Rahm, E.: Don't match twice: redundancy-free similarity computation with MapReduce. In: Data Analytics in the Cloud (2013)
References

Acknowledgements

We are thankful to the support provided by the following projects:

• FP7 ICT IdeaGarden STREP [http://idea-garden.org/](http://idea-garden.org/)

• GSRT ARISTEIA (LODGOV) Data Governance in the era of the Web of Data
License

These slides are made available under a Creative Commons Attribution-ShareAlike license (CC BY-SA 3.0):
http://creativecommons.org/licenses/by-sa/3.0/

You can share and remix this work, provided that you keep the attribution to the original authors intact, and that, if you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.