Entity Resolution in the Web of Data

Part II

Kostas Stefanidis[1], Vasilis Efthymiou[1,2],
Melanie Herschel[3,4], Vassilis Christophides[5]

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

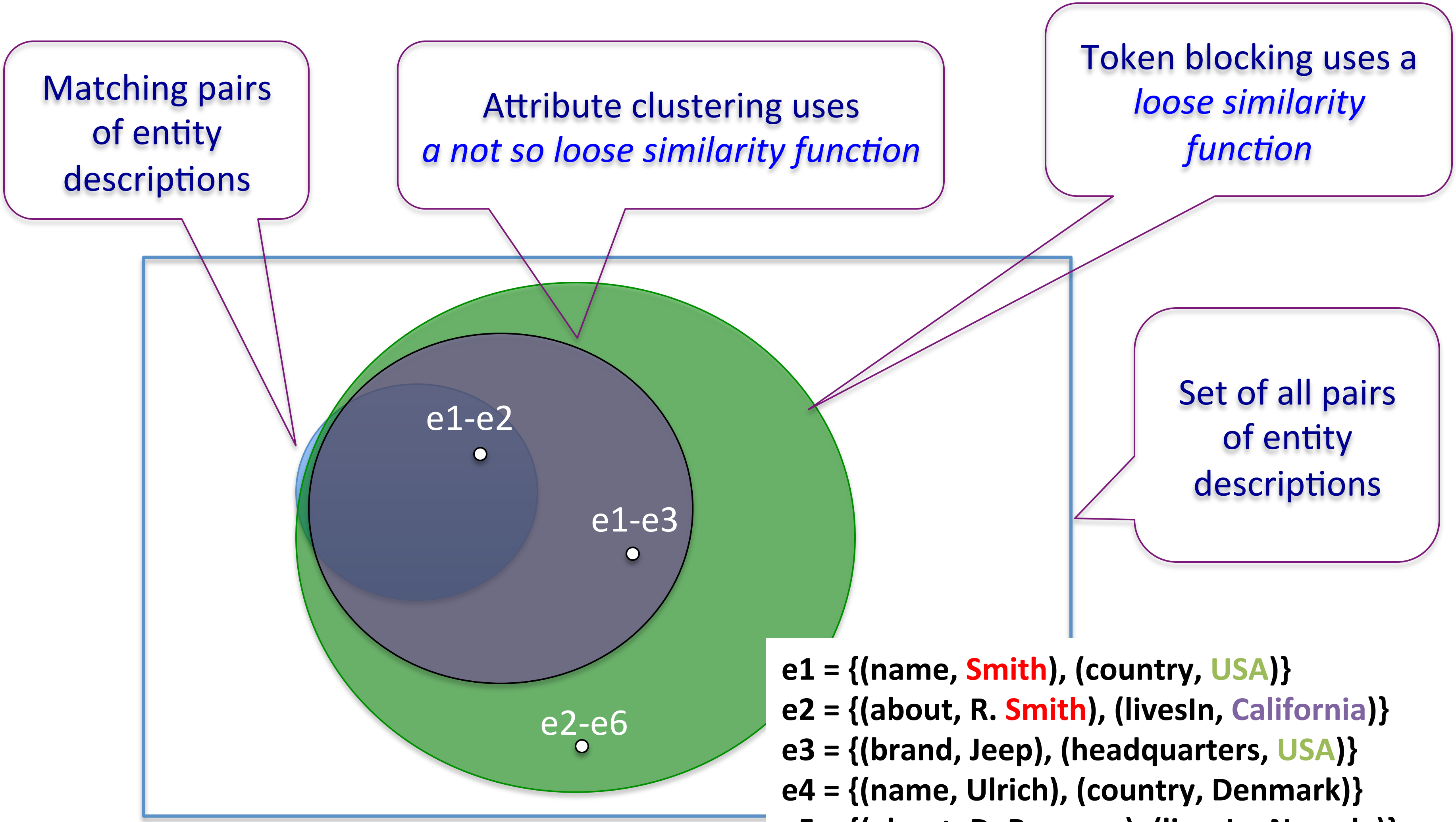
[1]FORTH, [2]University of Crete, [3]Université Paris Sud, [4]Inria Saclay,
[5]Paris R&I Center, Tehcnicolor

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

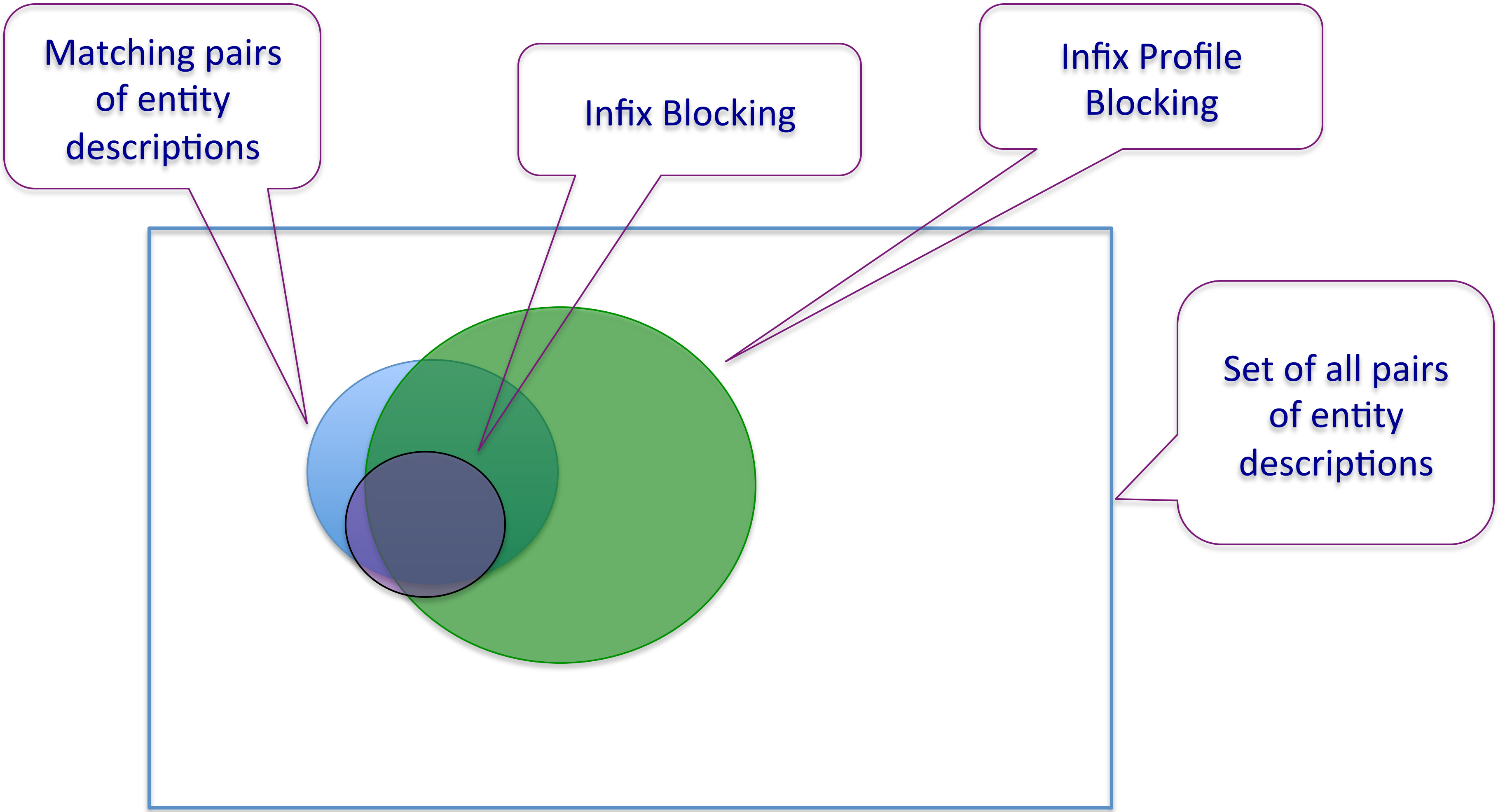
Attribute clustering uses *a not so loose similarity function*

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*

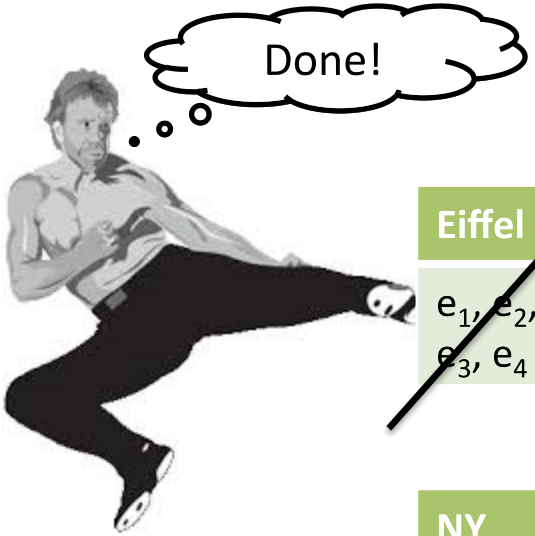
- Remove oversized blocks
  - Threshold on the number of descriptions in a block
- Order blocks
  - Examine first the blocks which are more likely to contain matches
- Remove low-order blocks
  - We do not gain much by examining them
- Order comparisons
  - Perform first the comparisons that are more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

Removing Oversized Blocks

Eiffel	e_1, e_2, e_3, e_4
Tower	e_1, e_4, e_5
Liberty	e_2, e_3
NY	e_2, e_3
Paris	e_1, e_4
1889	e_1, e_4

Block size threshold = 3

Removing Oversized Blocks



~~Eiffel~~
~~e_1, e_2, e_3, e_4~~

Tower
e_1, e_4, e_5

Liberty
e_2, e_3

NY
e_2, e_3

Paris
e_1, e_4

1889
e_1, e_4

Block size threshold = 3

Block Post-processing

The goal: *Reduce further the number of comparisons*

HOW?

- 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

Ordering Blocks [Papadakis et al. 2011(a)]

Assign a utility value to each block:

- $u(b_i) = gain(b_i) / cost(b_i)$

$gain(b_i)$: #superfluous comparisons spared in subsequently examined blocks

$cost(b_i)$: #comparisons entailed in b_i

Estimation for Clean-Clean Entity Resolution: $u(bi) \approx 1 / \max(|b_{i,1}|, |b_{i,2}|)$

$b_{i,j}$ 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]

$$\text{Match\_likelihood}(e_i, e_j) = \text{Jaccard}(\text{blocks}(e_i), \text{blocks}(e_j)) =$$
$$|\text{blocks}(e_i) \cap \text{blocks}(e_j)| \,/\, |\text{blocks}(e_i) \cup \text{blocks}(e_j)|$$

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	STEP 2	STEP 3
Block Building	Meta-Blocking	Block Post-Processing

Meta-blocking [Papadakis et al. 2013 (b)]

A generic procedure for block re-construction

- 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

name	**Eiffel Tower**
architect	Sauvestre
year	1889
location	Paris

e1

name	**Statue of Liberty**
architect	Bartholdi Eiffel
year	1886
located	NY

e2

about	**Lady liberty**
architect	Eiffel
location	NY

e3

about	**Eiffel Tower**
architect	Sauvestre
year	1889
located	Paris

e4

name	**White Tower**
location	Thessaloniki
year-constructed	1450

e5

Blocks:

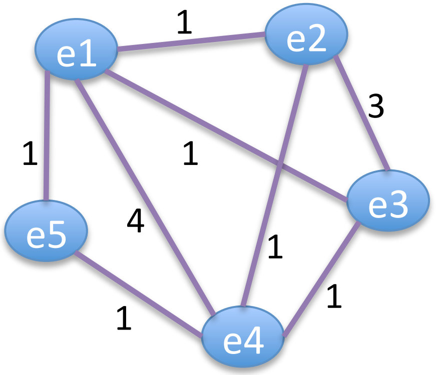
(with token blocking)

Eiffel	**Tower**	**Liberty**
e_1, e_2, e_3, e_4	e_1, e_4, e_5	e_2, e_3

NY	**Paris**	**1889**
e_2, e_3	e_1, e_4	e_1, e_4

13 comparisons
to identify 2 matches

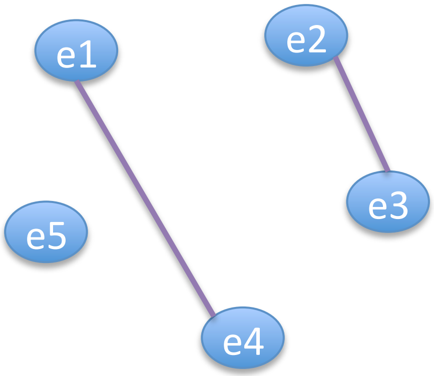
Blocking graph:



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

	Name	Year	Architects	Location
e_1	Eiffel Tower	1889	Sauvestre	Paris
e_2	Statue of Liberty	1886	Bartholdi, Eiffel	NY
e_3	Lady Liberty	1885	Eiffel	Liberty Island, NY
e_4	Eiffel Tower	1889		Paris
e_5	Miss Liberty	1886	Gustave Eiffel	Liberty Island

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

1889	1886	1885
e_1, e_4	e_2, e_5	e_3

P	N	L
e_1, e_4	e_2	e_3, e_5

Iterative Blocking - Example

	Name	**Year**	**Architects**	**Location**
e_1	Eiffel Tower	1889	Sauvestre	Paris
e_2	Statue of Liberty	1886	Bartholdi, Eiffel	NY
e_3	Lady Liberty	1885	Eiffel	Liberty Island, NY
e_4	Eiffel Tower	1889		Paris
e_5	Miss Liberty	1886	Gustave Eiffel	Liberty Island

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

1889	**1886**	**1885**
e_1, e_4	e_2, e_5	e_3

P	**N**	**L**
e_1, e_4	e_2	e_3, e_5

e_1, e_4 match! they are merged as e_{14}

Iterative Blocking - Example

	Name	Year	Architects	Location
e_1	Eiffel Tower	1889	Sauvestre	Paris
e_2	Statue of Liberty	1886	Bartholdi, Eiffel	NY
e_3	Lady Liberty	1885	Eiffel	Liberty Island, NY
e_4	Eiffel Tower	1889		Paris
e_5	Miss Liberty	1886	Gustave Eiffel	Liberty Island

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

1889	1886	1885
e_1, e_4, e_{14}	e_2, e_5	e_3

P	N	L
e_1, e_4, e_{14}	e_2	e_3, e_5

Iterative Blocking - Example

	Name	Year	Architects	Location
e_1	Eiffel Tower	1889	Sauvestre	Paris
e_2	Statue of Liberty	1886	Bartholdi, Eiffel	NY
e_3	Lady Liberty	1885	Eiffel	Liberty Island, NY
e_4	Eiffel Tower	1889		Paris
e_5	Miss Liberty	1886	Gustave Eiffel	Liberty Island

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

1889	1886	1885
e_1, e_4, e_{14}	e_2, e_5	e_3

P	N	L
e_1, e_4, e_{14}	e_2	e_3, e_5

e_2, e_5 match! they are merged as e_{25}

Iterative Blocking - Example

	Name	Year	Architects	Location
e_1	Eiffel Tower	1889	Sauvestre	Paris
e_2	Statue of Liberty	1886	Bartholdi, Eiffel	NY
e_3	Lady Liberty	1885	Eiffel	Liberty Island, NY
e_4	Eiffel Tower	1889		Paris
e_5	Miss Liberty	1886	Gustave Eiffel	Liberty Island

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

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

P	N	L
e_1, e_4, e_{14}	e_2, e_{25}	e_3, e_5, e_{25}

e_2, e_5 match! they are merged as e_{25}

Iterative Blocking - Example

	Name	Year	Architects	Location
e_1	Eiffel Tower	1889	Sauvestre	Paris
e_2	Statue of Liberty	1886	Bartholdi, Eiffel	NY
e_3	Lady Liberty	1885	Eiffel	Liberty Island, NY
e_4	Eiffel Tower	1889		Paris
e_5	Miss Liberty	1886	Gustave Eiffel	Liberty Island

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

P
e_1, e_4, e_{14}

N
e_2, e_{25}

L
e_3, e_5, e_{25}

Iterative Blocking - Example

	Name	Year	Architects	Location
e_1	Eiffel Tower	1889	Sauvestre	Paris
e_2	Statue of Liberty	1886	Bartholdi, Eiffel	NY
e_3	Lady Liberty	1885	Eiffel	Liberty Island, NY
e_4	Eiffel Tower	1889		Paris
e_5	Miss Liberty	1886	Gustave Eiffel	Liberty Island

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

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

P	N	L
e_{14}	e_2, e_{25}	e_3, e_5, e_{25}

Iterative Blocking - Example

	Name	Year	Architects	Location
e_1	Eiffel Tower	1889	Sauvestre	Paris
e_2	Statue of Liberty	1886	Bartholdi, Eiffel	NY
e_3	Lady Liberty	1885	Eiffel	Liberty Island, NY
e_4	Eiffel Tower	1889		Paris
e_5	Miss Liberty	1886	Gustave Eiffel	Liberty Island

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

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

P	N	L
e_{14}	e_{25}	e_3, e_5, e_{25}

Iterative Blocking - Example

	Name	Year	Architects	Location
e_1	Eiffel Tower	1889	Sauvestre	Paris
e_2	Statue of Liberty	1886	Bartholdi, Eiffel	NY
e_3	Lady Liberty	1885	Eiffel	Liberty Island, NY
e_4	Eiffel Tower	1889		Paris
e_5	Miss Liberty	1886	Gustave Eiffel	Liberty Island

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

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

P	N	L
e_{14}	e_{25}	e_3, e_{25}

e_3, e_{25} match! they are merged as e_{235}

Iterative Blocking - Example

	Name	Year	Architects	Location
e_1	Eiffel Tower	1889	Sauvestre	Paris
e_2	Statue of Liberty	1886	Bartholdi, Eiffel	NY
e_3	Lady Liberty	1885	Eiffel	Liberty Island, NY
e_4	Eiffel Tower	1889		Paris
e_5	Miss Liberty	1886	Gustave Eiffel	Liberty Island

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}, e_{235}
1885	e_3, e_{235}
P	e_{14}
N	e_{25}, e_{235}
L	e_3, e_{25}, e_{235}

e_3, e_{25} match! they are merged as e_{235}

Iterative Blocking - Example

	Name	Year	Architects	Location
e_1	Eiffel Tower	1889	Sauvestre	Paris
e_2	Statue of Liberty	1886	Bartholdi, Eiffel	NY
e_3	Lady Liberty	1885	Eiffel	Liberty Island, NY
e_4	Eiffel Tower	1889		Paris
e_5	Miss Liberty	1886	Gustave Eiffel	Liberty Island

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

1889	1886	1885
e_{14}	e_2, e_5, e_{25}, e_{235}	e_3, e_{235}

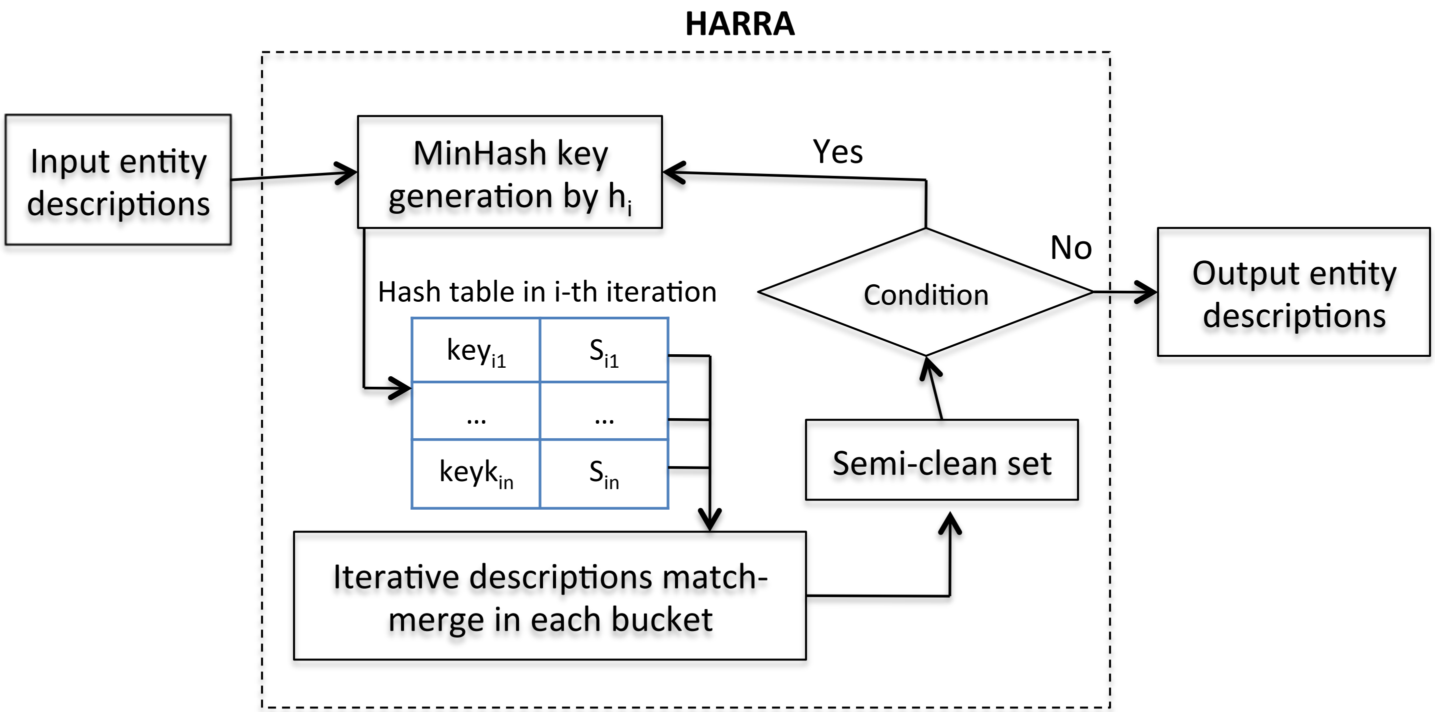
P	N	L
e_{14}	e_{25}, e_{235}	e_3, e_{25}, e_{235}

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)



HARRA

Input entity descriptions

MinHash key generation by h_i

Hash table in i-th iteration

key_{i1}	S_{i1}
…	…
$keyk_{in}$	S_{in}

Iterative descriptions match-merge in each bucket

Semi-clean set

Condition

Yes

No

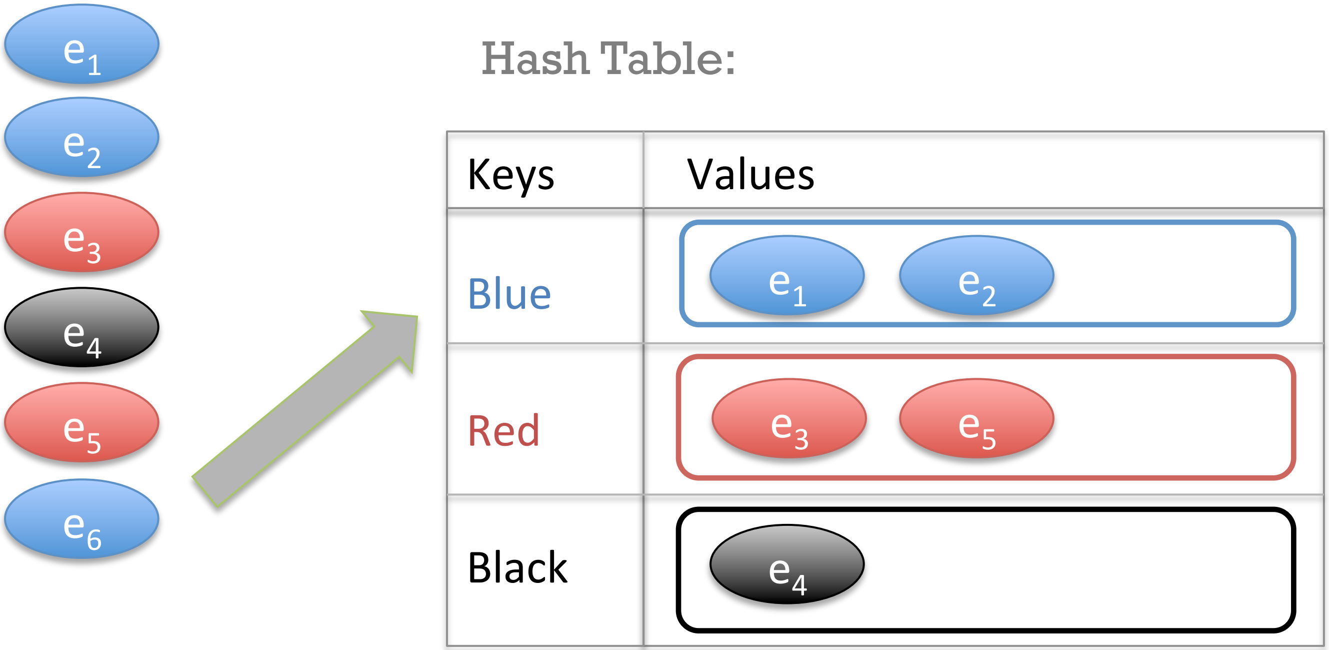
Output entity descriptions

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:

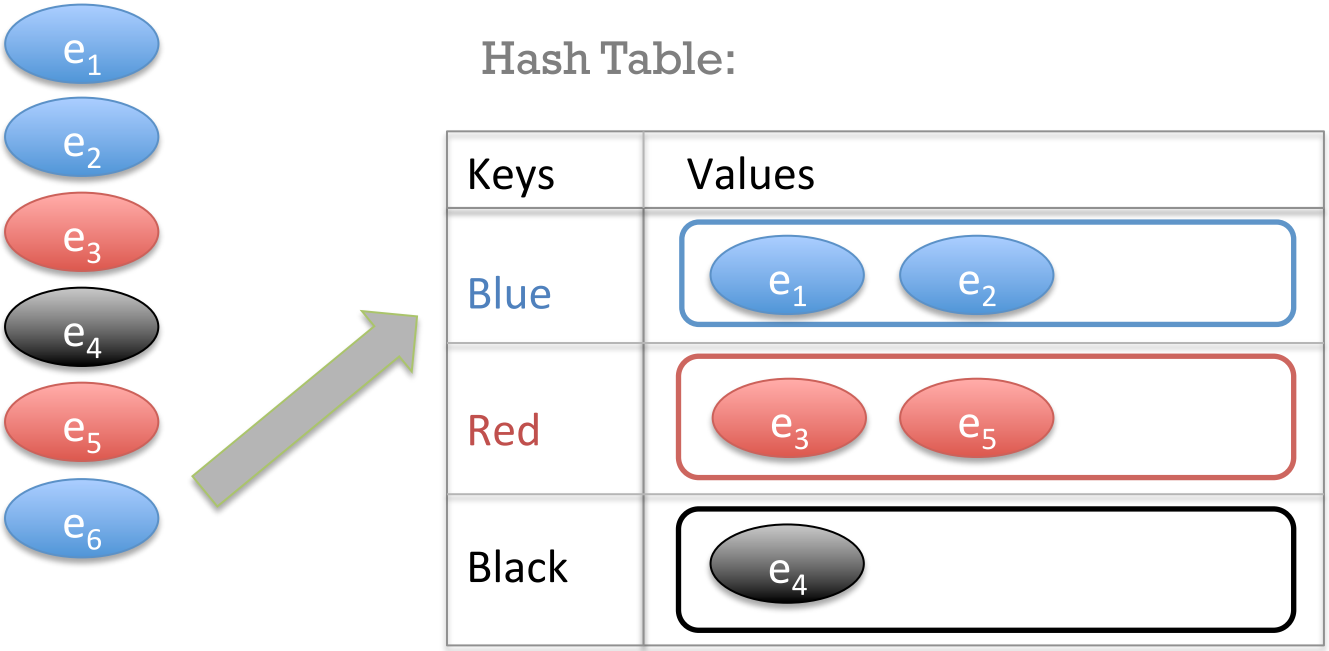
Keys	Values
Blue	e_1 e_2
Red	e_3 e_5
Black	e_4

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



Hash Table:

Keys	Values
Blue	e_1 e_2
Red	e_3 e_5
Black	e_4

HARRA - Example

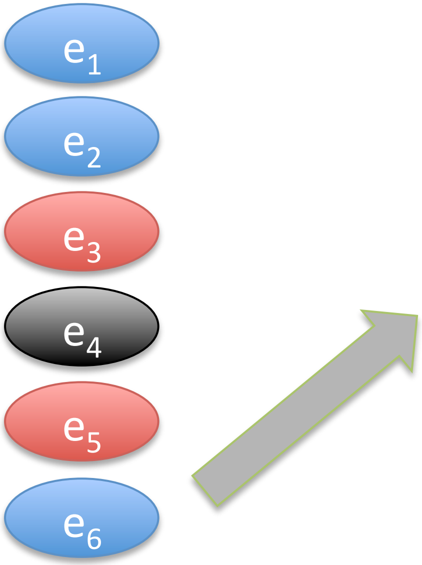
Before placing it there, we check if it matches e_1 or e_2

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:

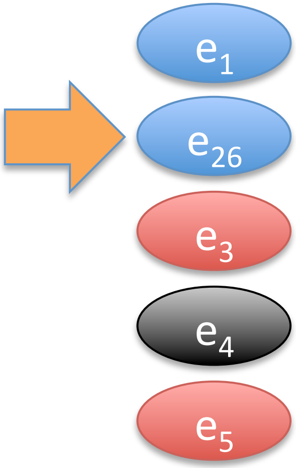
Keys	Values
Blue	e_1 e_{26}
Red	e_3 e_5
Black	e_4

HARRA - Example

Continue until:

- no merge occurs, OR
- saved comparisons > threshold, OR
- # iterations > constant

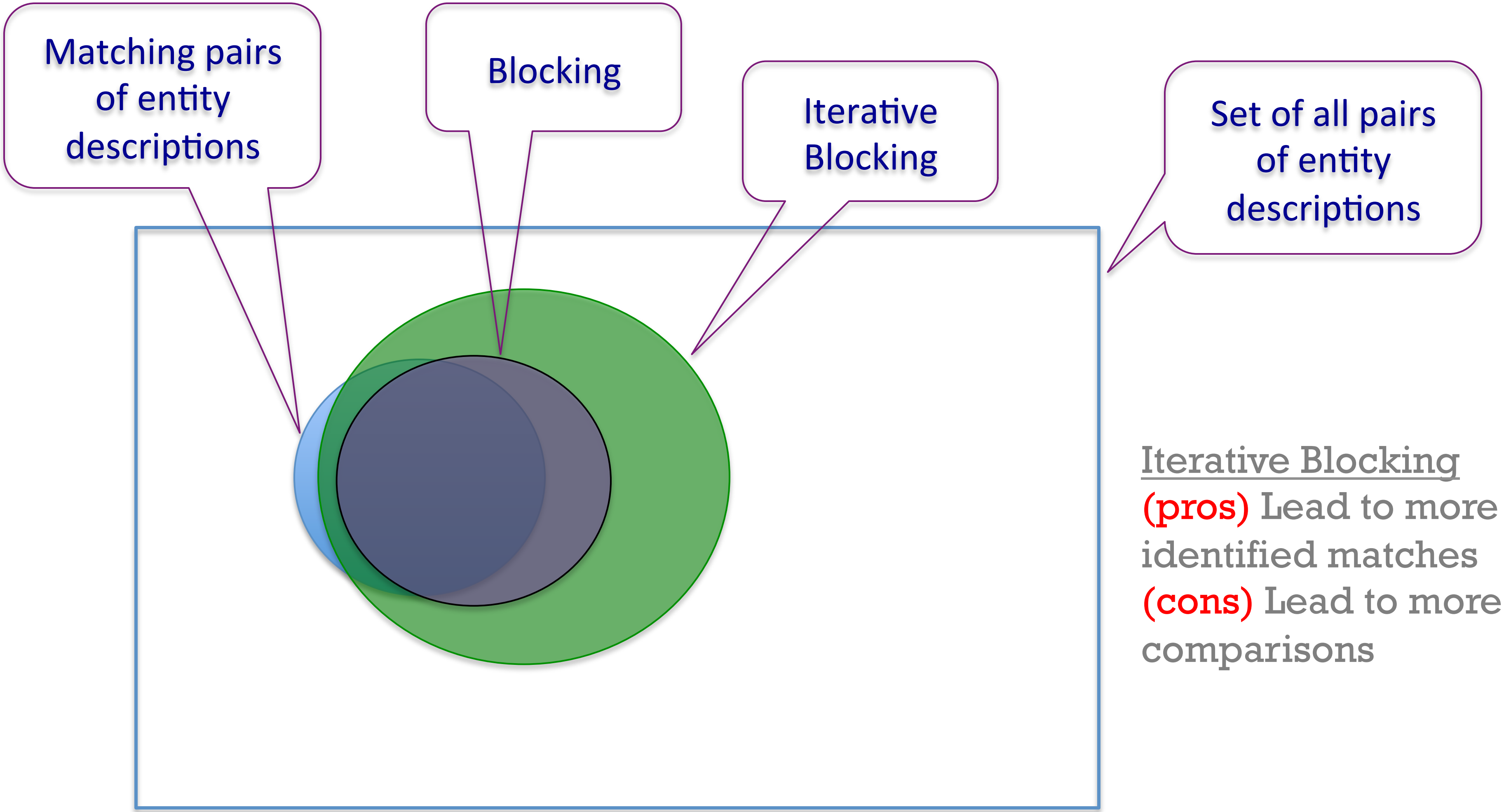
Re-initialize the input:



Hash Table:

Keys	Values
Blue	
Red	
Black	

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!

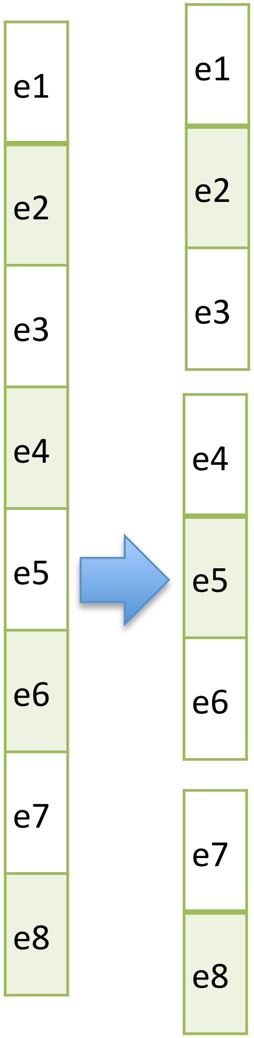
The map phase reflects blocking (re-distribute descriptions)

The reduce phase reflects entity resolution (check for matches)

MapReduce – Input Data

e1
e2
e3
e4
e5
e6
e7
e8

MapReduce – Input Data Partitioning



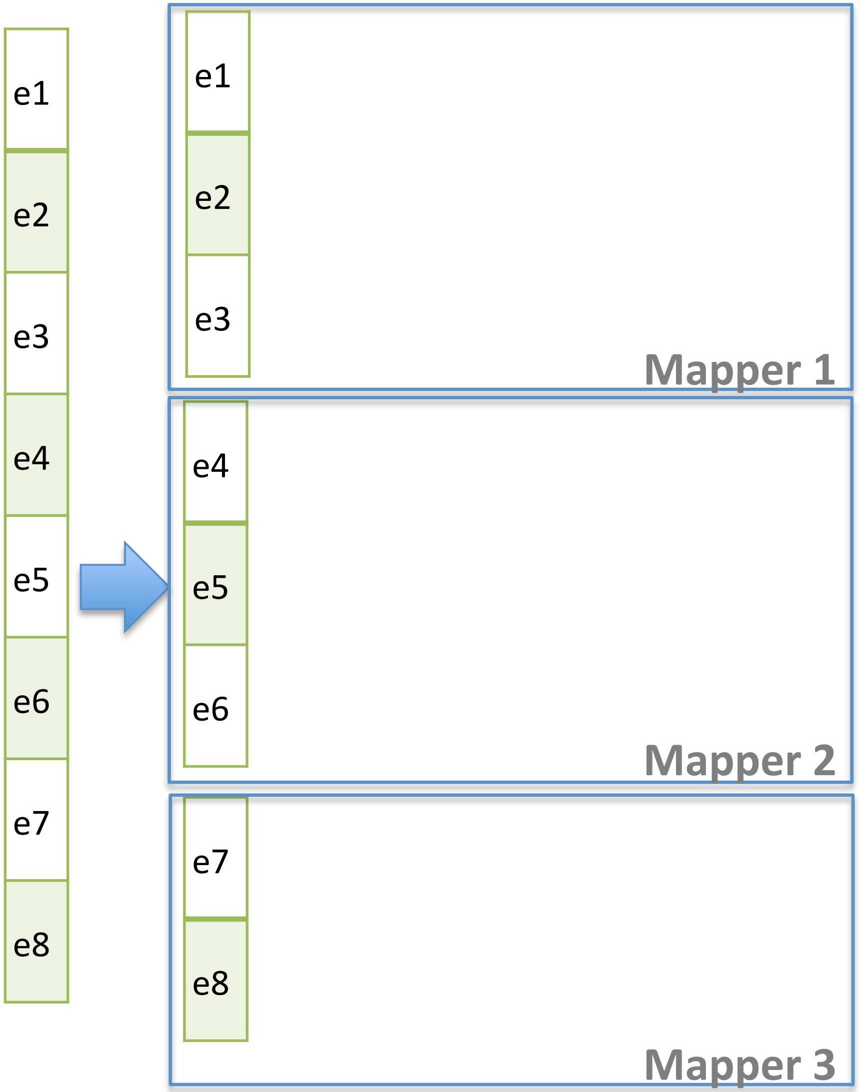
e1
e2
e3
e4
e5
e6
e7
e8

e1
e2
e3

e4
e5
e6

e7
e8

MapReduce – Mapper Input



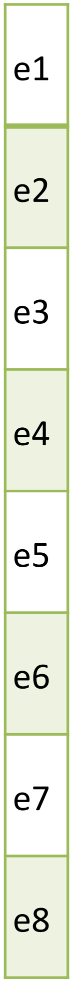
e1
e2
e3
e4
e5
e6
e7
e8

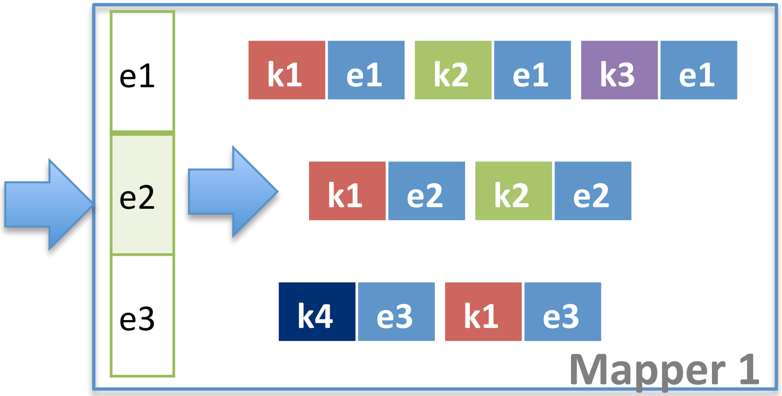
e1
e2
e3
Mapper 1

e4
e5
e6
Mapper 2

e7
e8
Mapper 3

MapReduce – Mapper Example





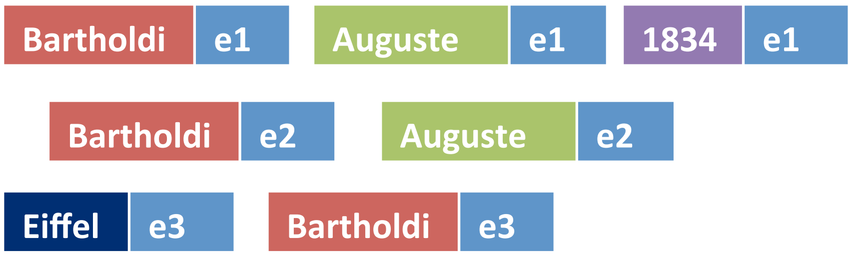
Input :

e1={(name, Auguste Bartholdi),(year,1834)}

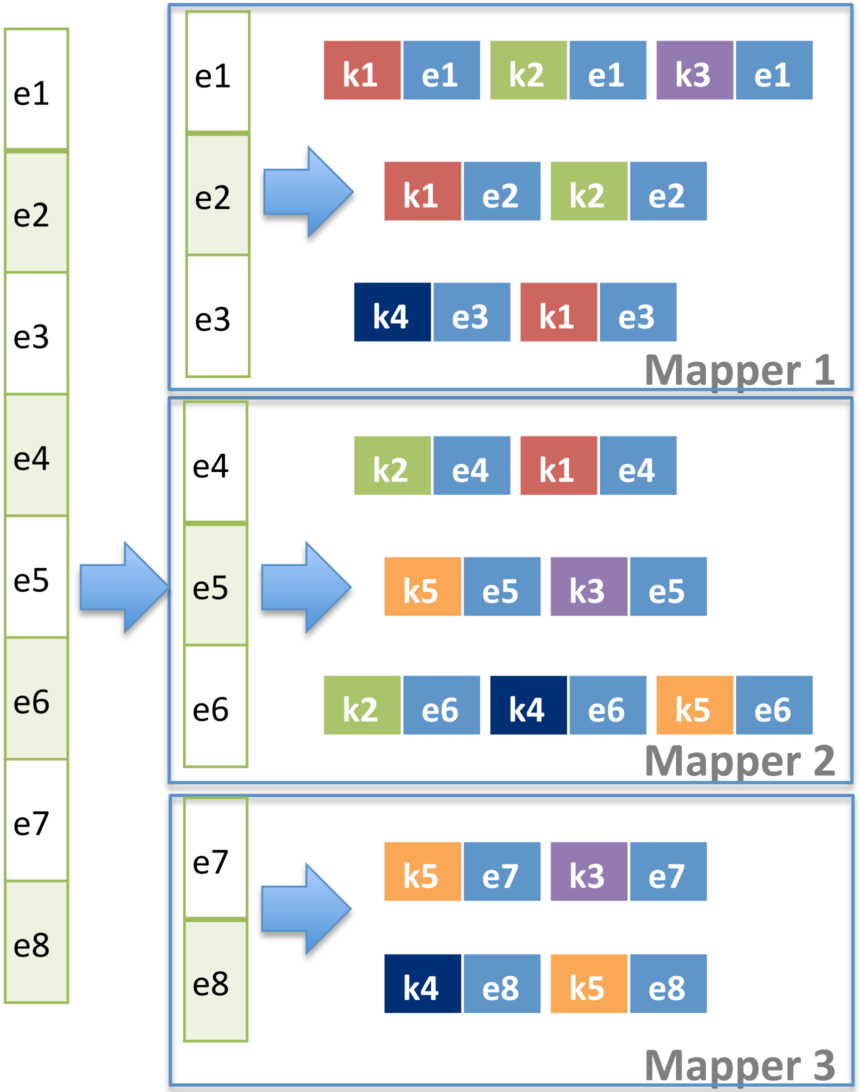
e2={(about, Auguste Bartholdi)}

e3={(architects, Bartholdi Eiffel)

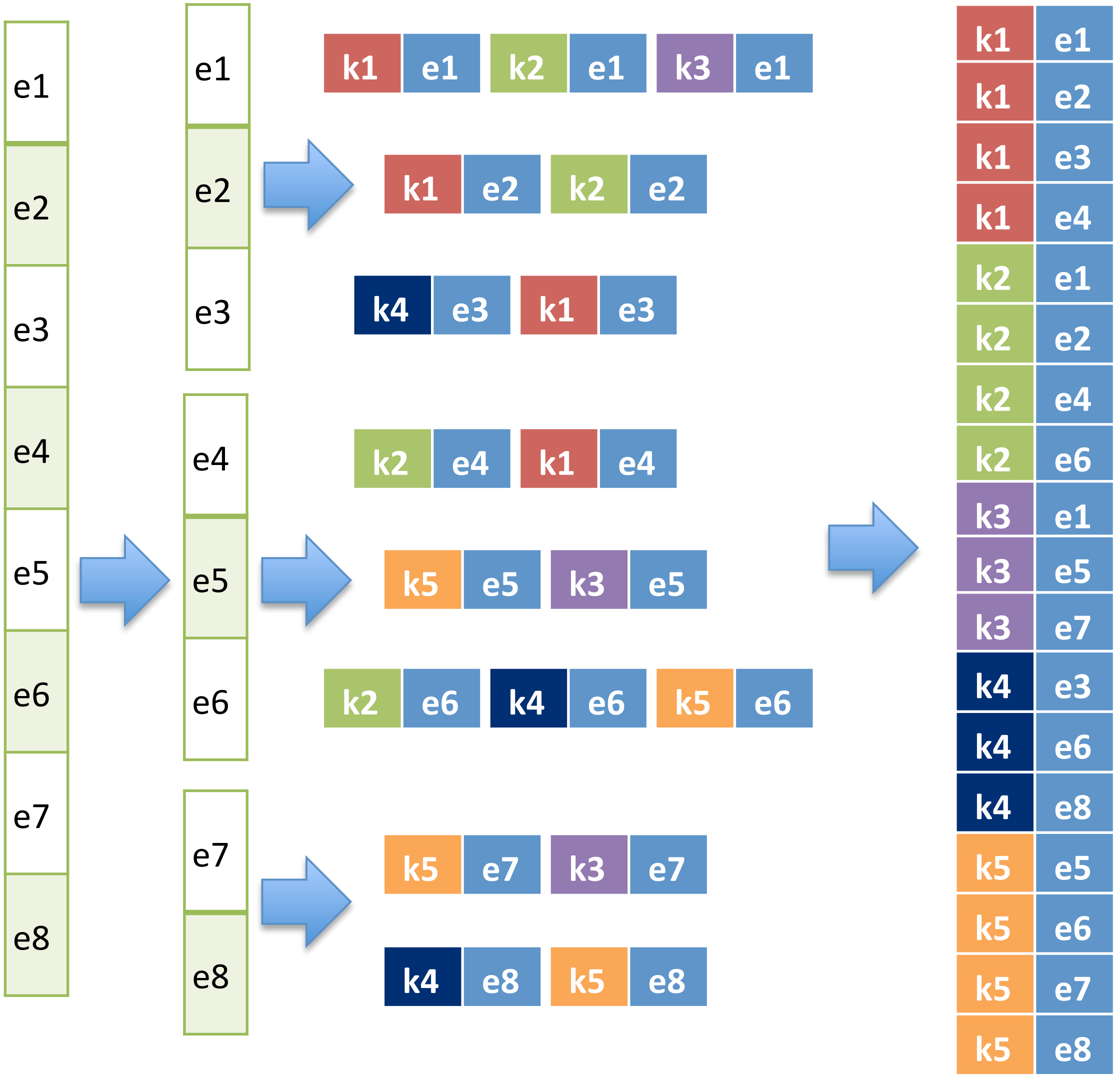
Output :



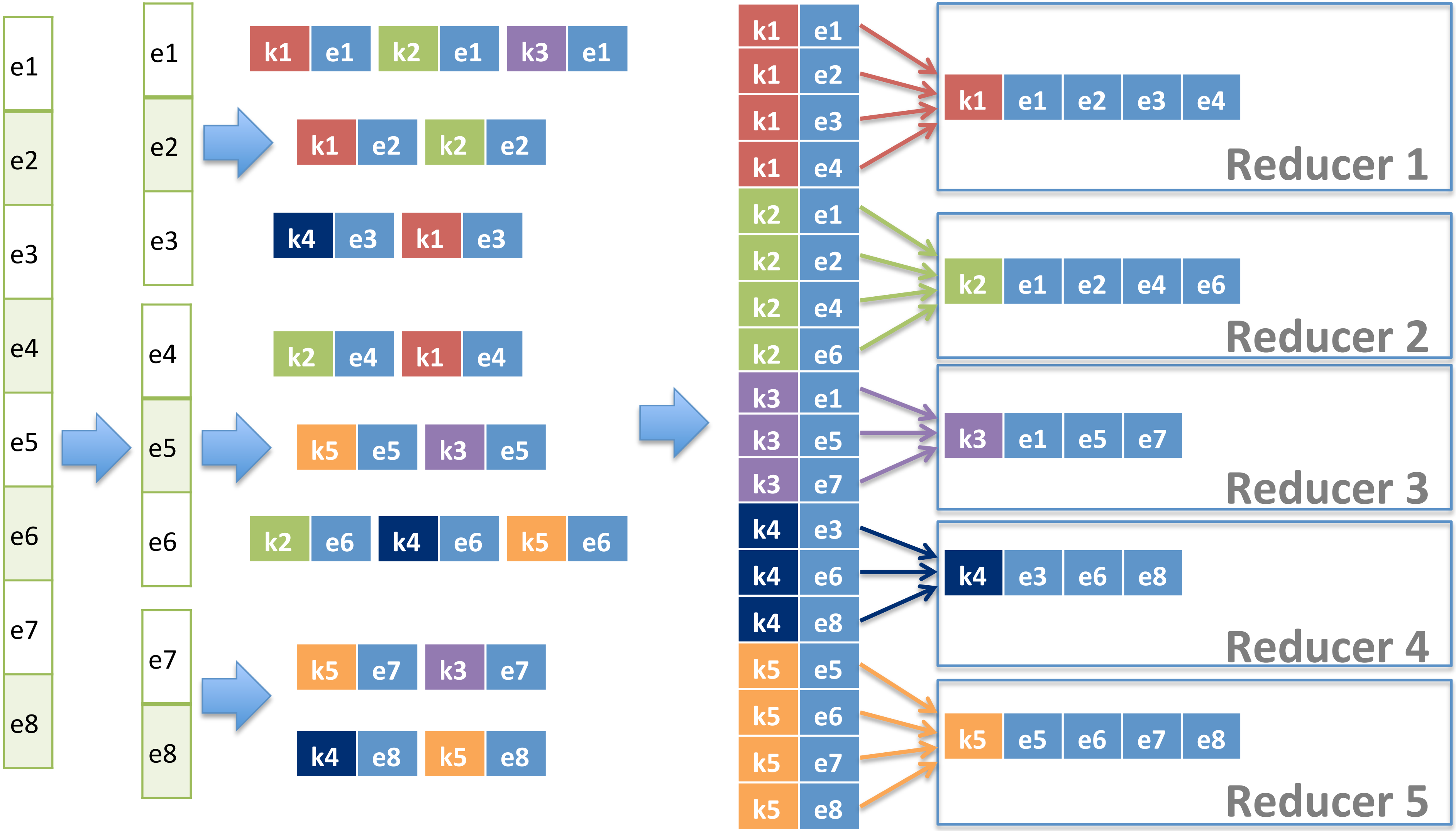
MapReduce – Mapper Output



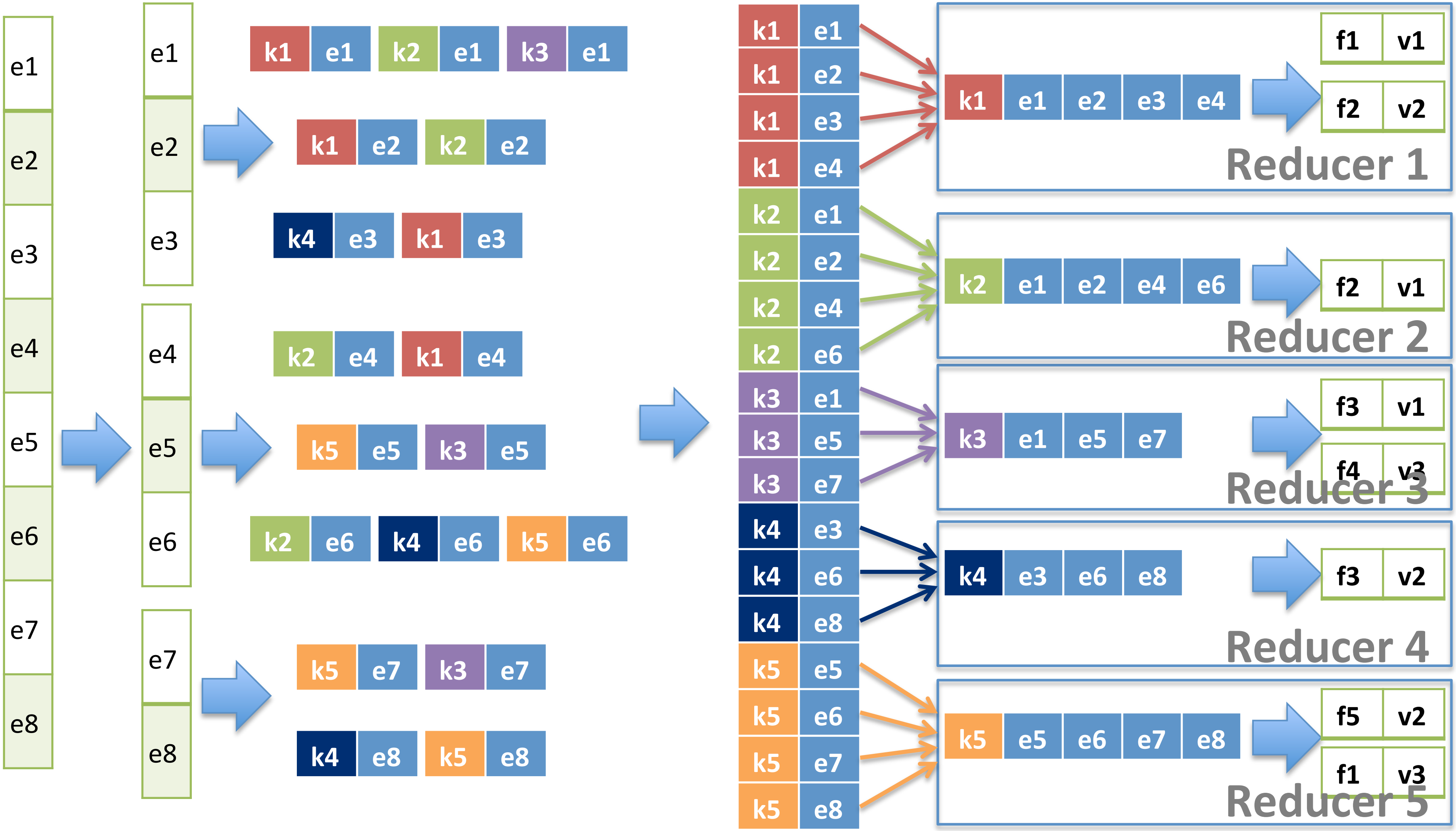
MapReduce – Shuffling & Sorting



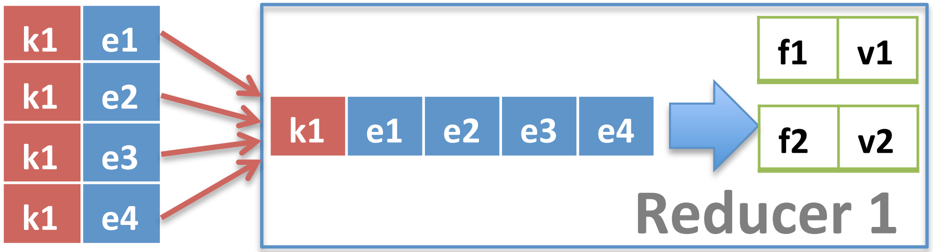
MapReduce – Merging

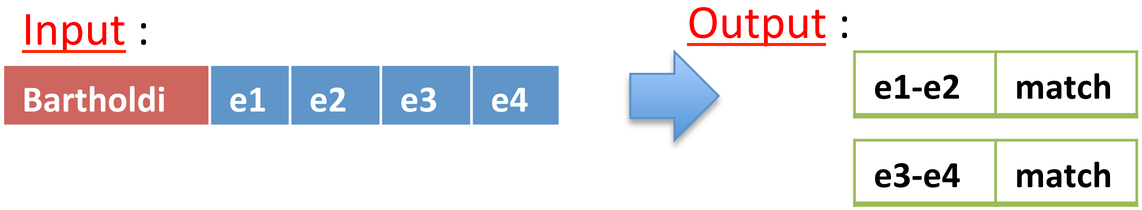


MapReduce – Reducer



MapReduce – Reducer Example





Input :

Bartholdi	e1	e2	e3	e4

Output :

e1-e2	match
e3-e4	match

Dedoop – Standard Blocking [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

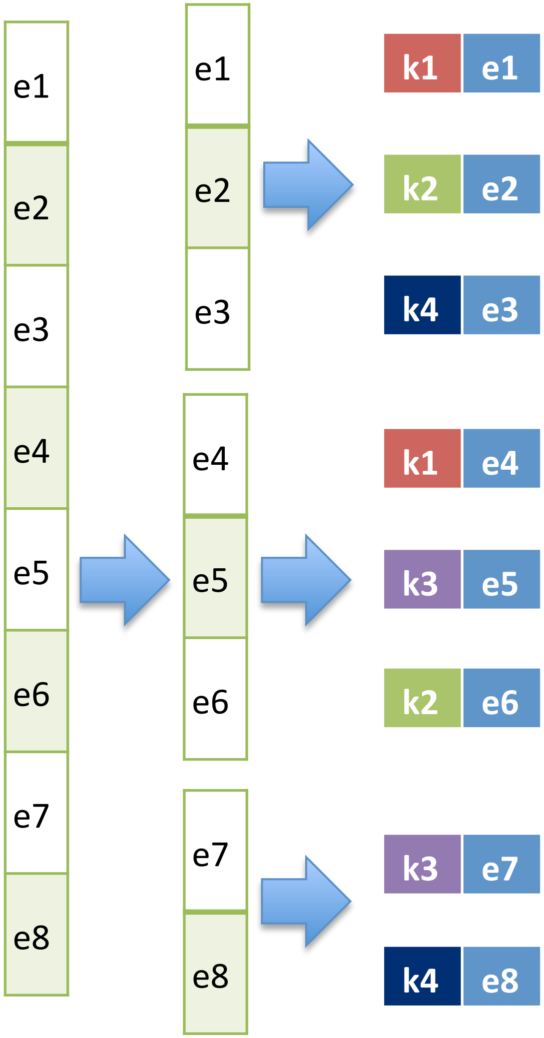
The partitioning operates on the BKVs and distributes (key, value) pairs among reduce tasks

- All entities sharing the same BKV are assigned to the same reduce task

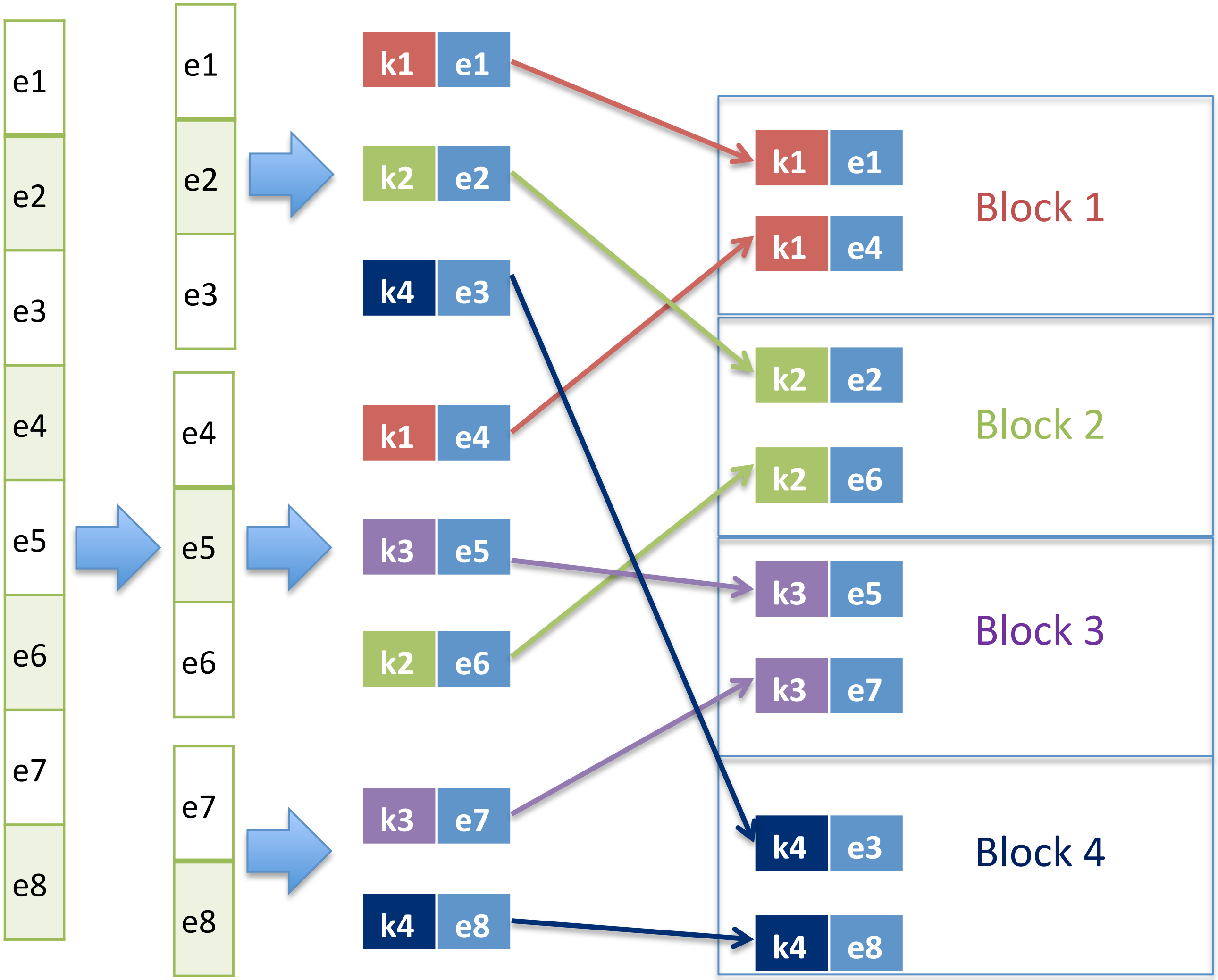
Reduce function: Computes in each block the similarities between all description pairs within the block

- Input: A BKV along with descriptions with this BKV
- Output: (key, value) pairs
    - key: a pair of descriptions
    - value: match/non-match

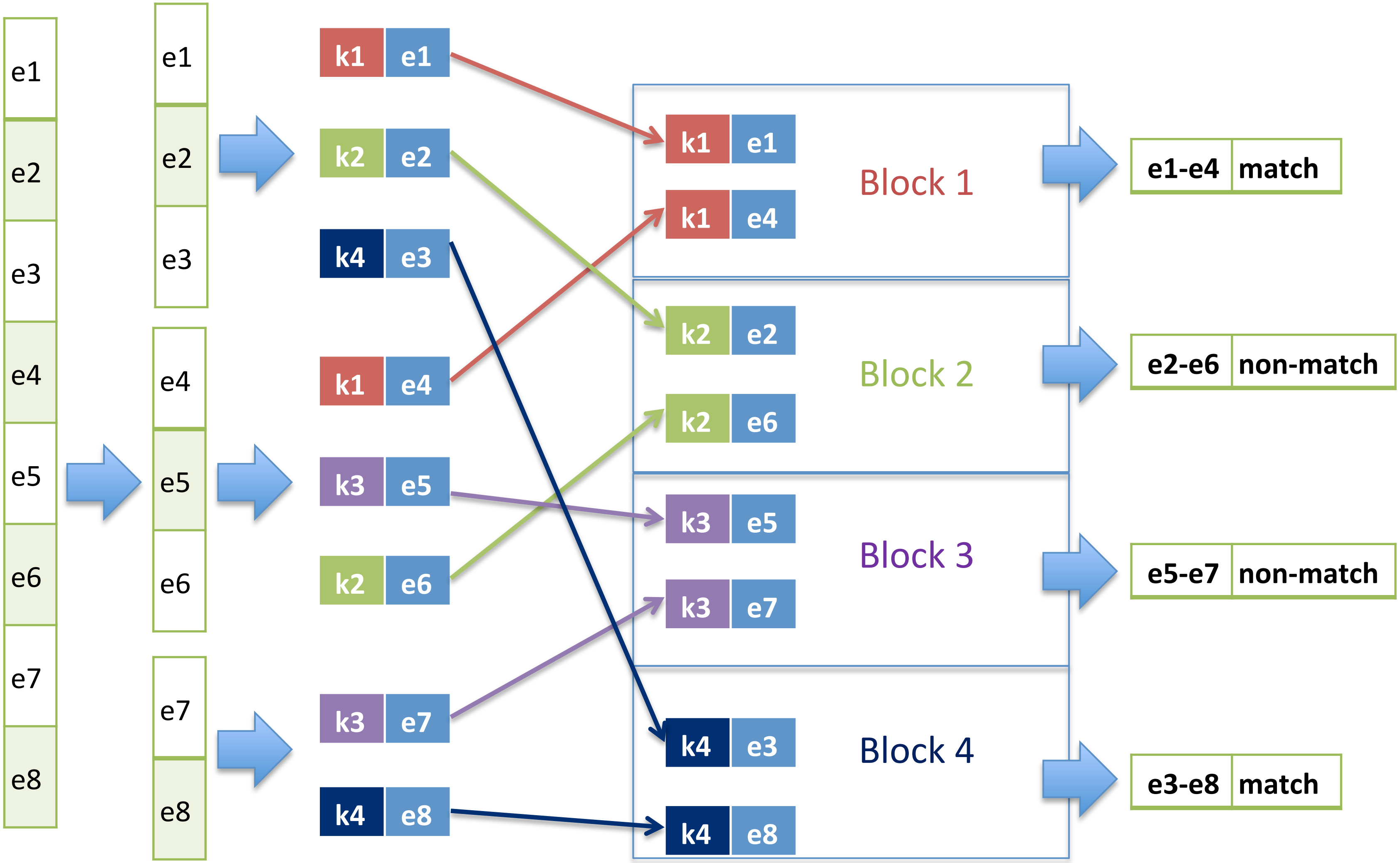
Dedoop – Mapper: BKVs as intermediate keys



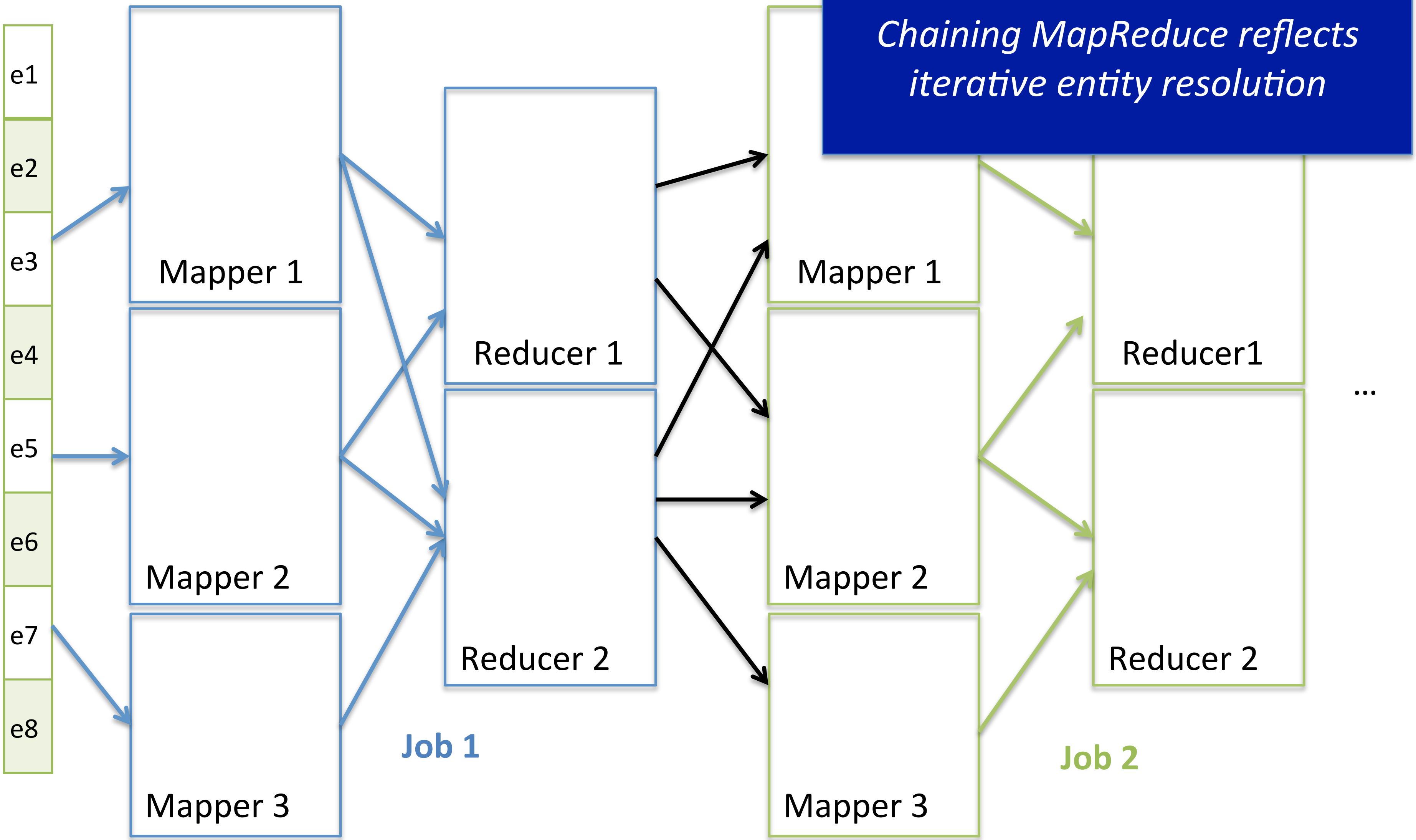
Dedoop – Mappers: Build Blocks



Dedoop – Reducers: Compare Block Contents



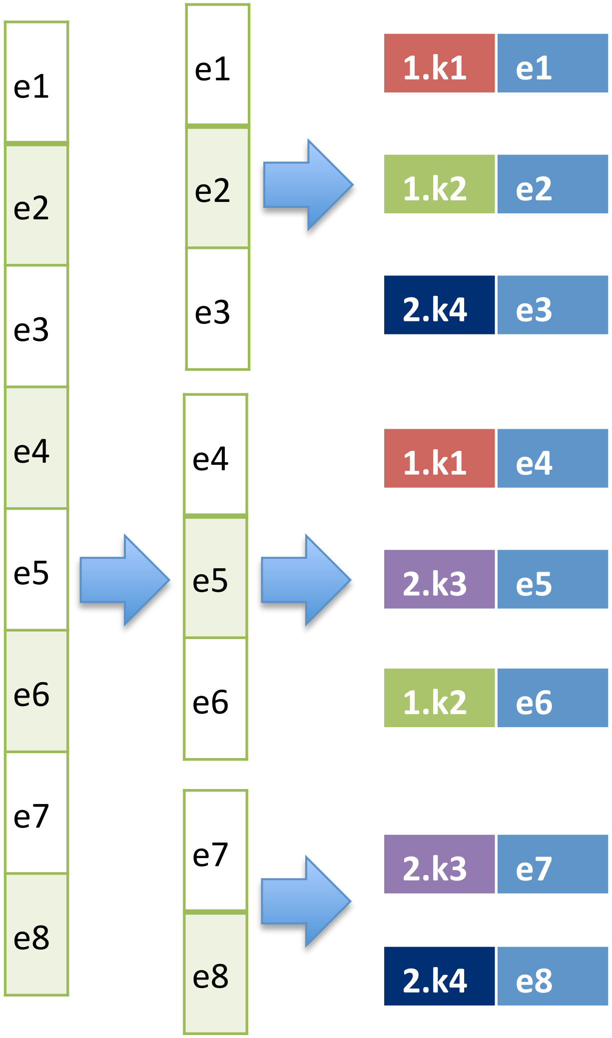
Chaining MapReduce Jobs



Chaining MapReduce reflects iterative entity resolution

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

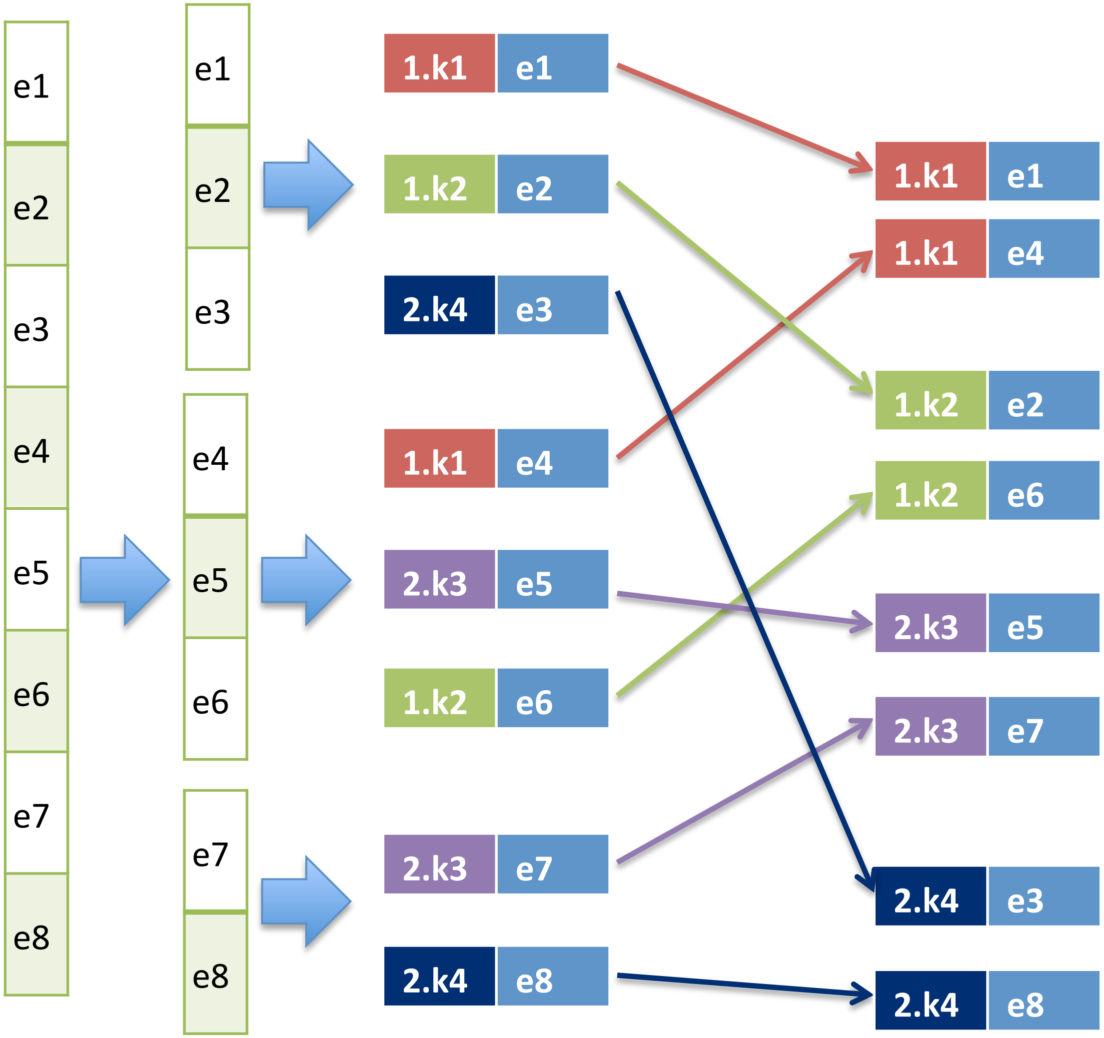
Dedoop – Sorted Neighborhood [Kolb et al. 2011]



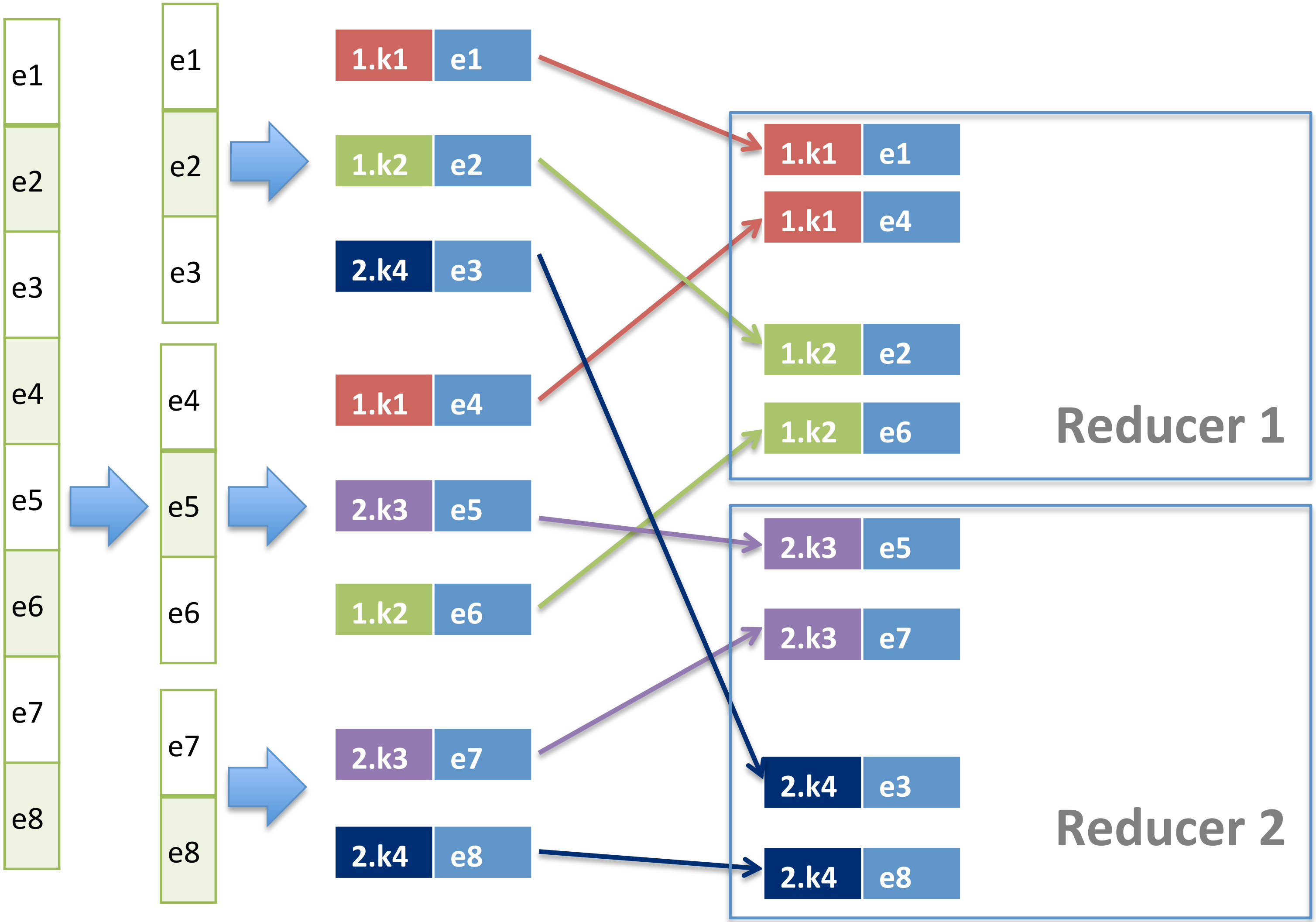
composite key = (partitionID, BKV)
partitionID(BKV) = 1, if BKV < "k3"
partitionID(BKV) = 2, else

(we know that we have two reducers available)

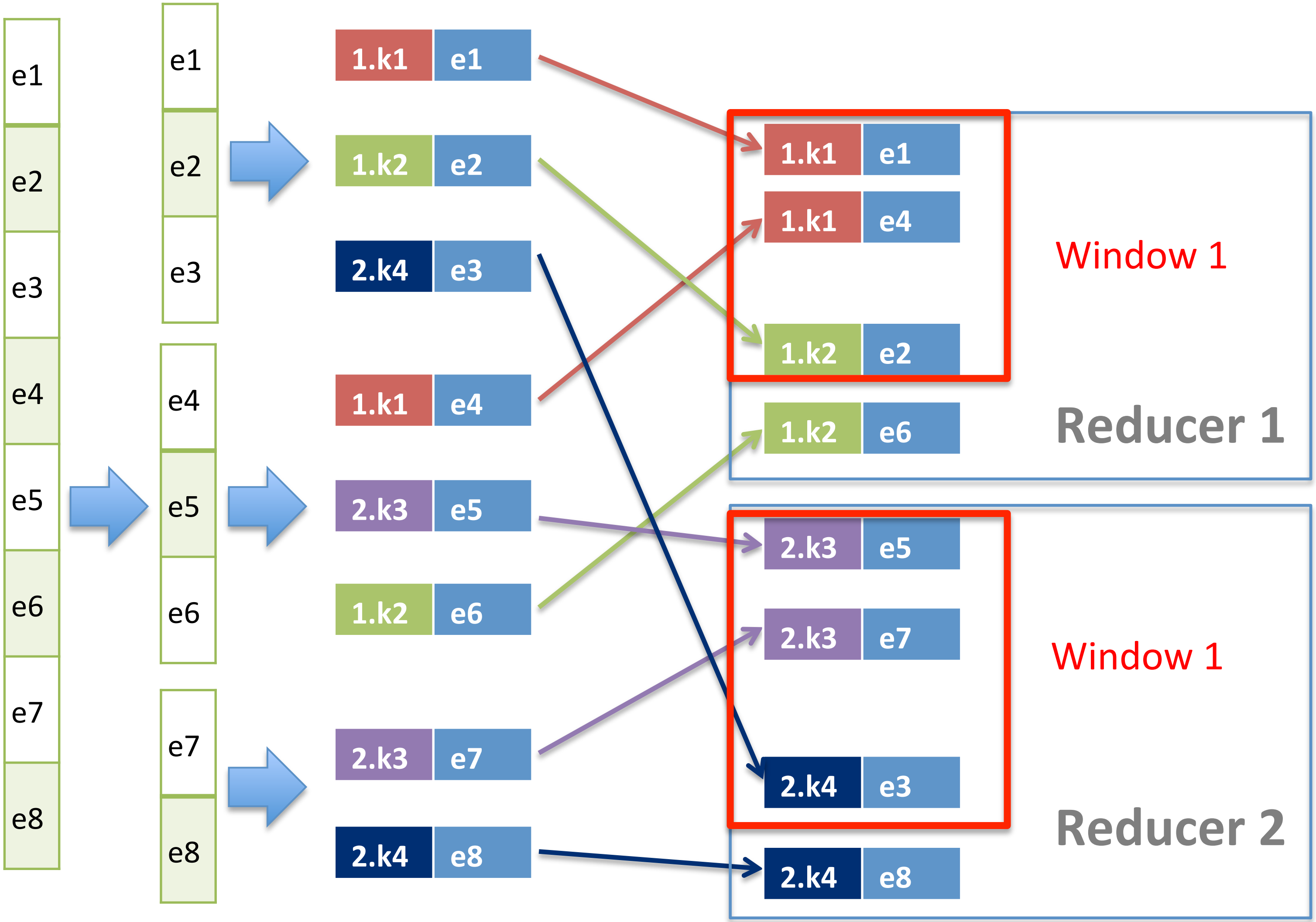
Dedoop SN: Sorting the Keys



Dedoop SN: Reducers Apply the Sliding Window



Dedoop SN: Reducers Apply the Sliding Window



e1 e2 e3 e4 e5 e6 e7 e8

e1 e2 e3

e4 e5 e6

e7 e8

1.k1 e1

1.k2 e2

2.k4 e3

1.k1 e4

2.k3 e5

1.k2 e6

2.k3 e7

2.k4 e8

1.k1 e1

1.k1 e4

1.k2 e2

1.k2 e6

Window 1

Reducer 1

2.k3 e5

2.k3 e7

2.k4 e3

2.k4 e8

Window 1

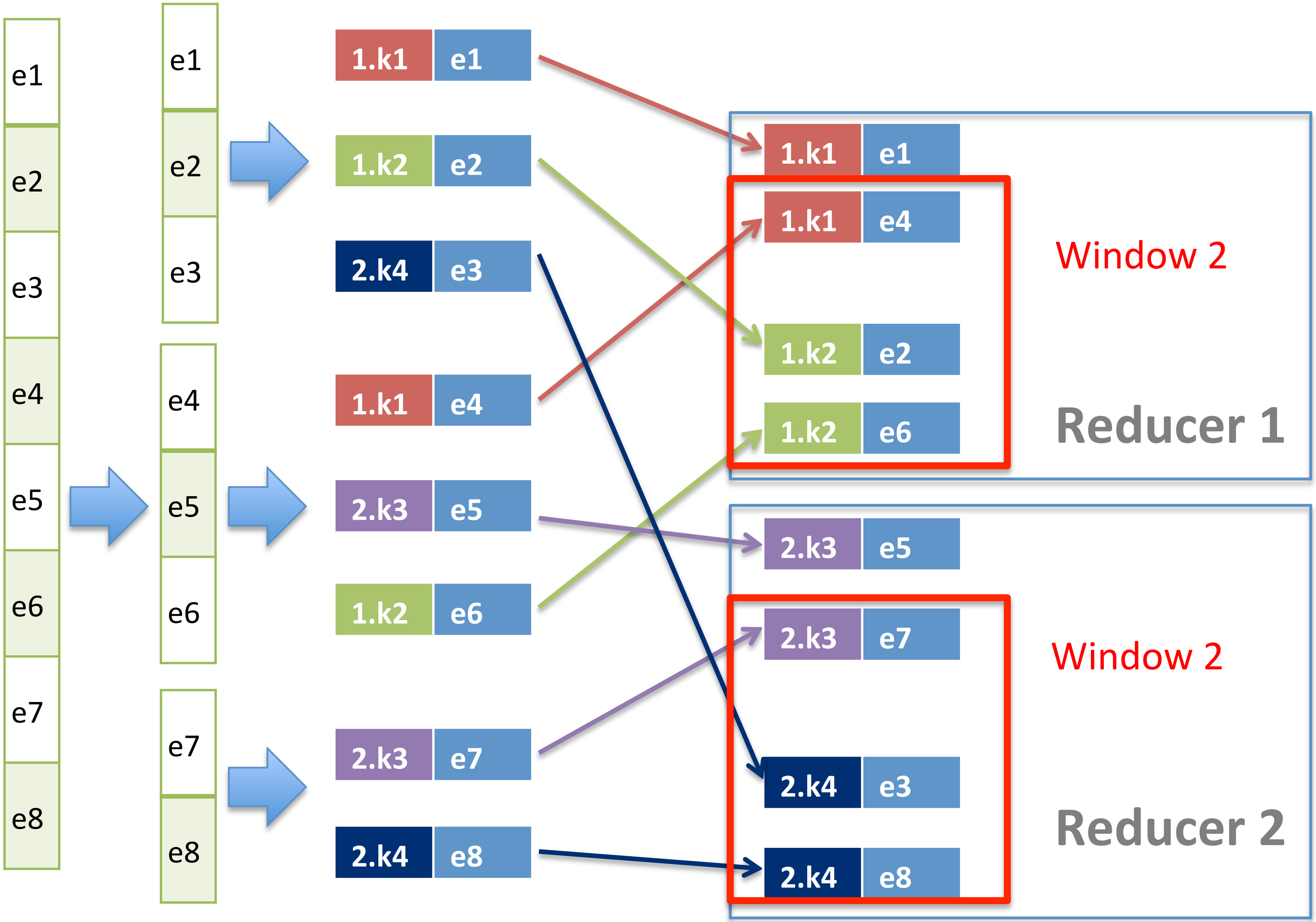
Reducer 2

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

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

w = 3

Dedoop SN: Reducers Apply the Sliding Window

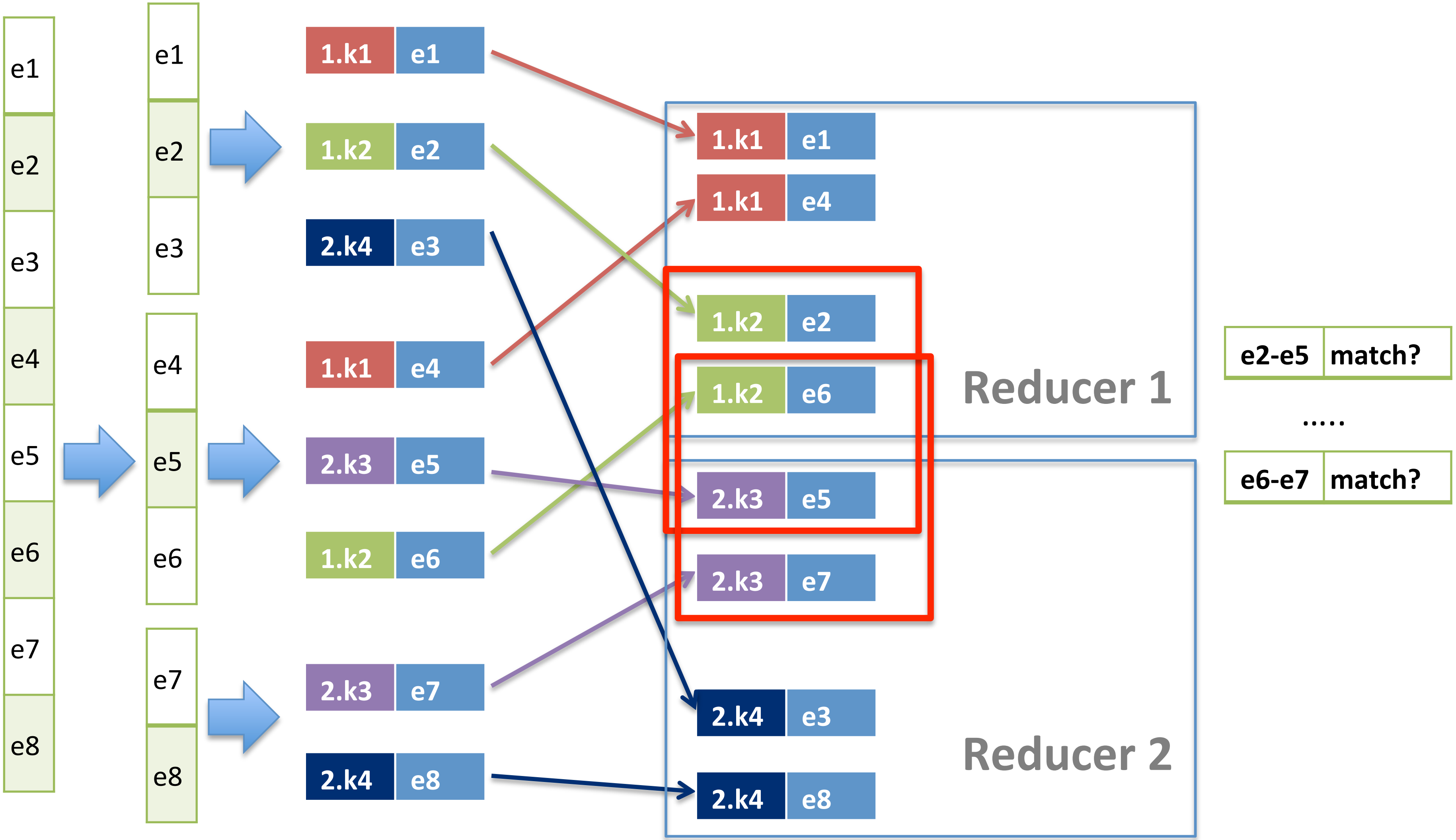


e4-e2	non-match
e4-e6	match
e2-e6	non-match

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

w = 3

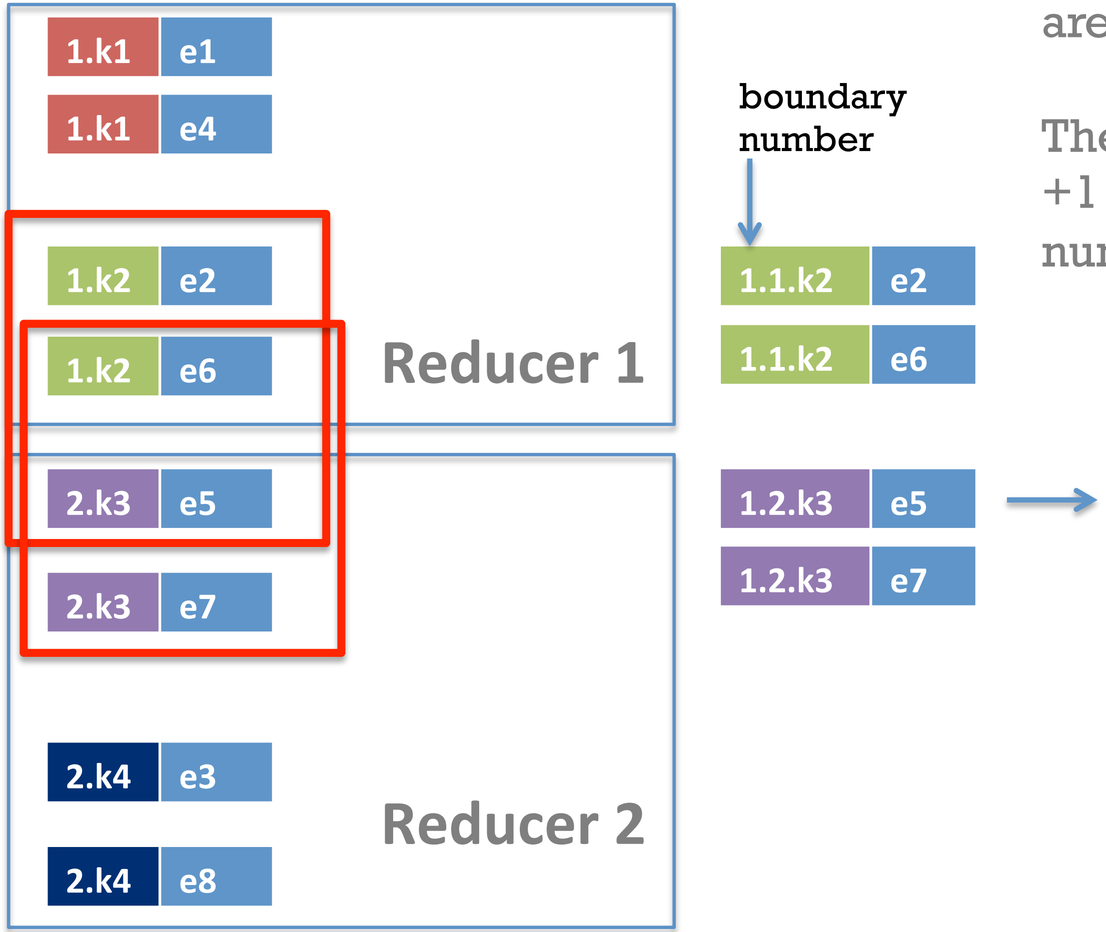
Dedoop SN: We Also Need To Compare The Boundary Entities



w = 3

Dedoop SN: Reducers Also Output the Boundary Descriptions

Add a boundary number prefix
to the output composite keys



Boundary number:

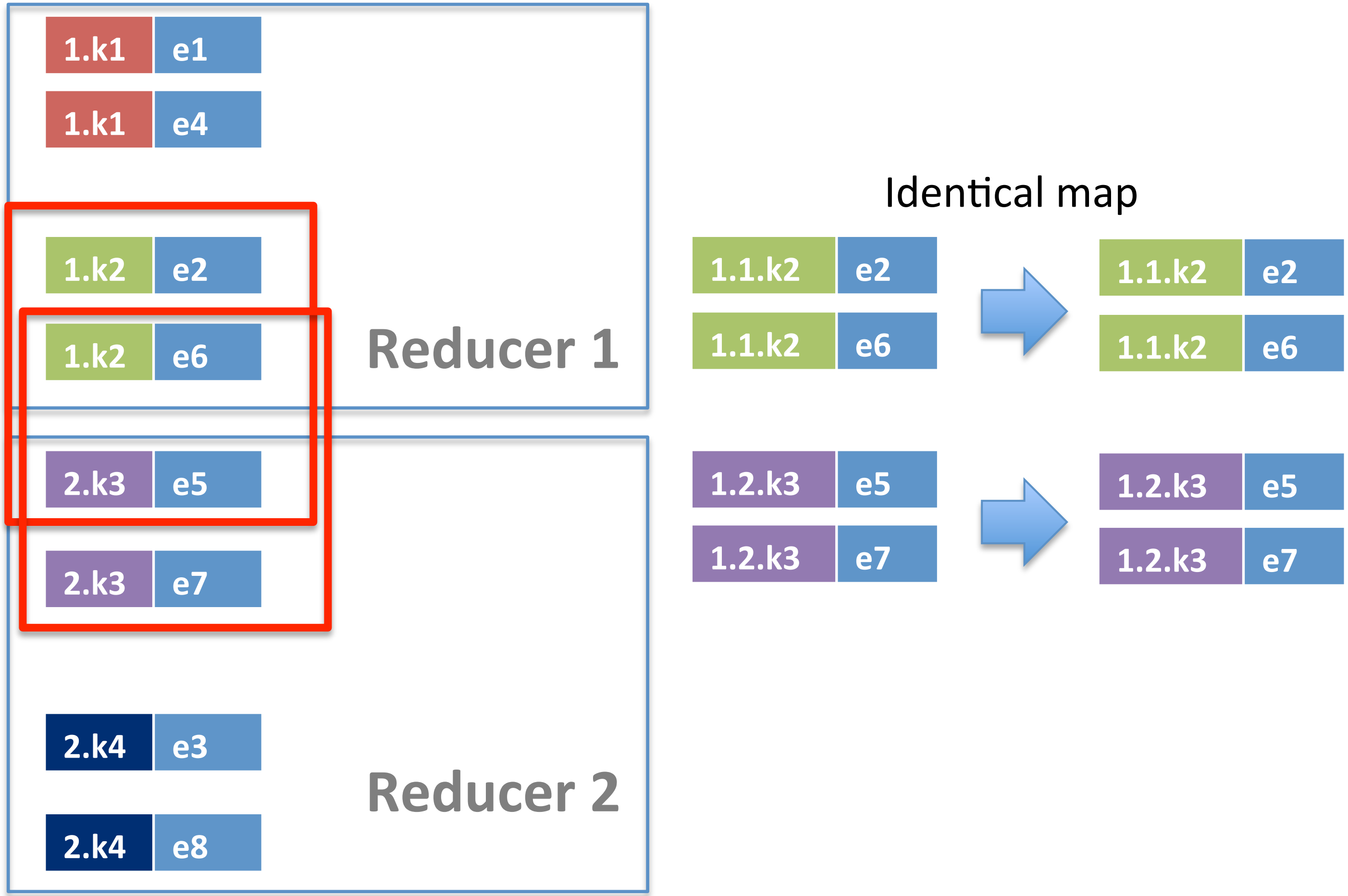
The last w-1 descriptions of reducer i are assigned the boundary number i

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

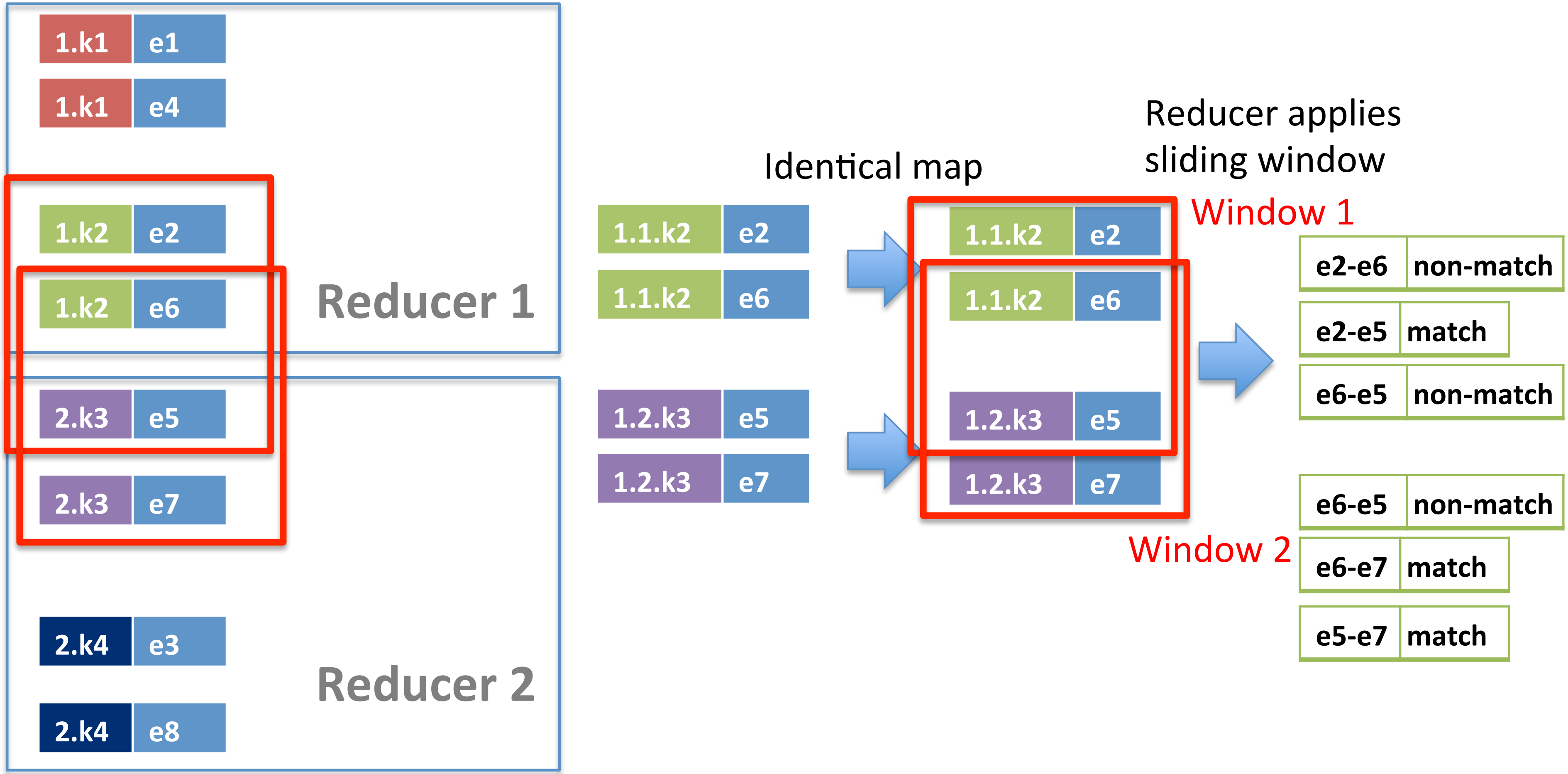
The actual blocking key of e5 is k3, it was assigned to reducer 2 and it is associated with boundary number 1

w = 3

Dedoop SN: New MapReduce Job for the Boundary Pairs

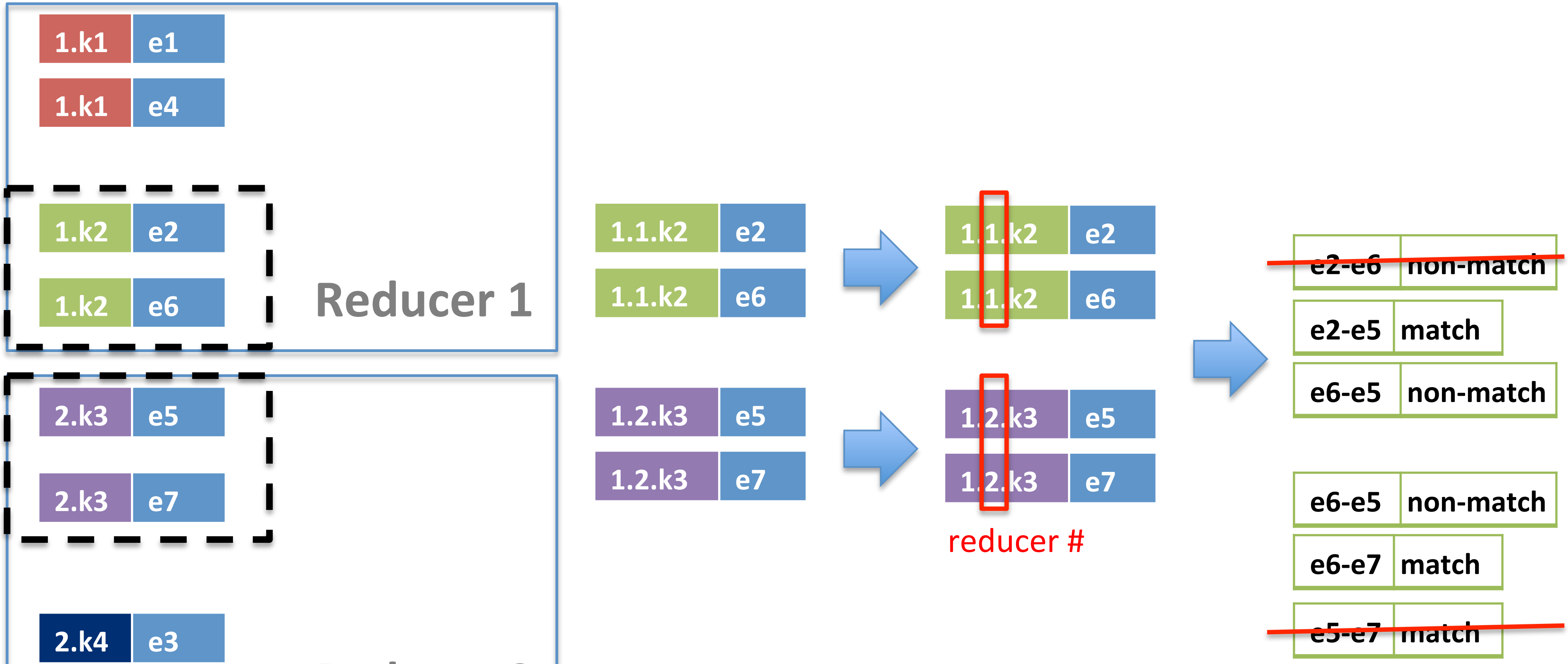


Dedoop SN: Partition by Boundary Number



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)

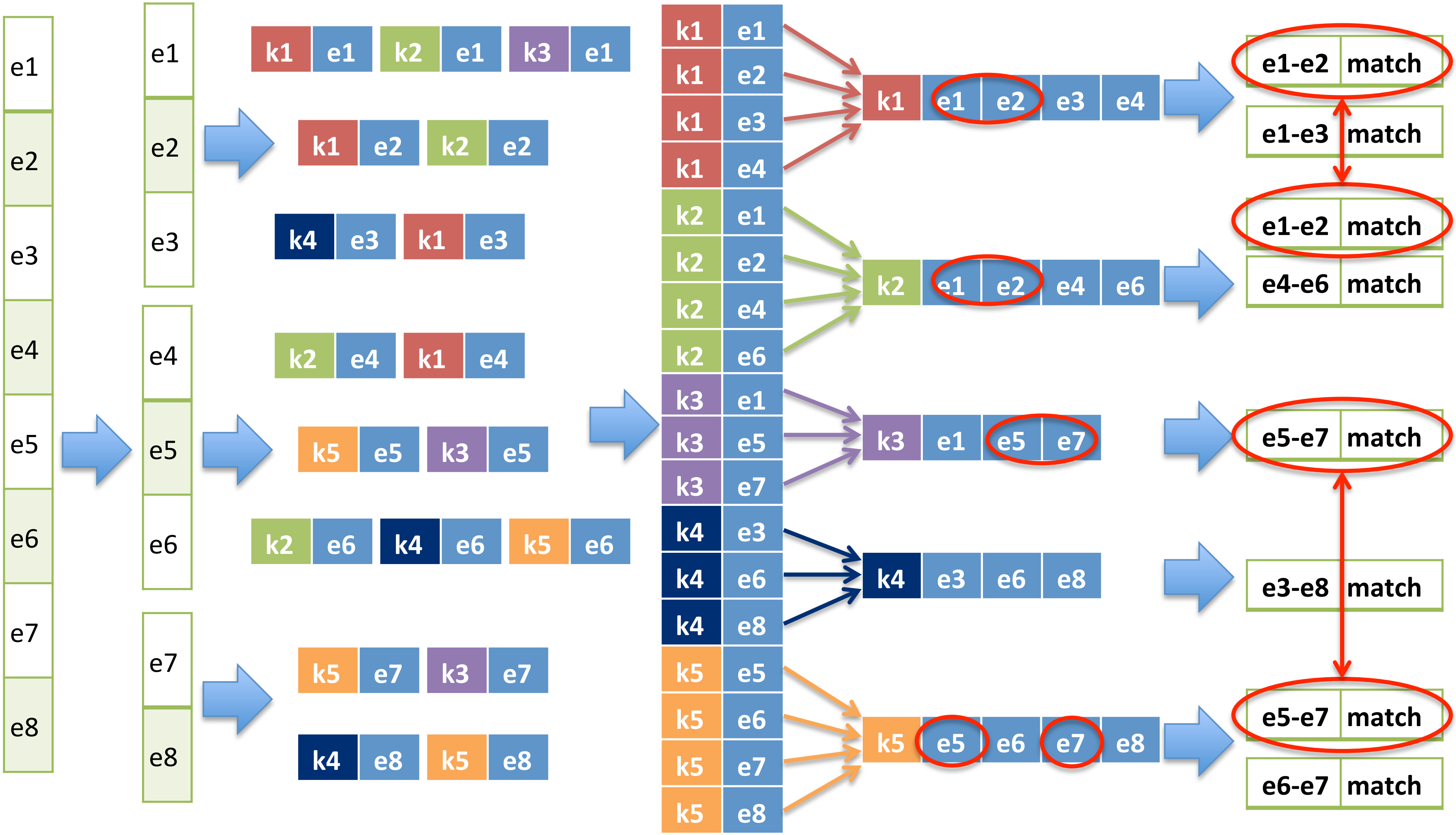
Don’t match twice [Kolb et al. 2013]

Overlapping blocks lead to repeated comparisons

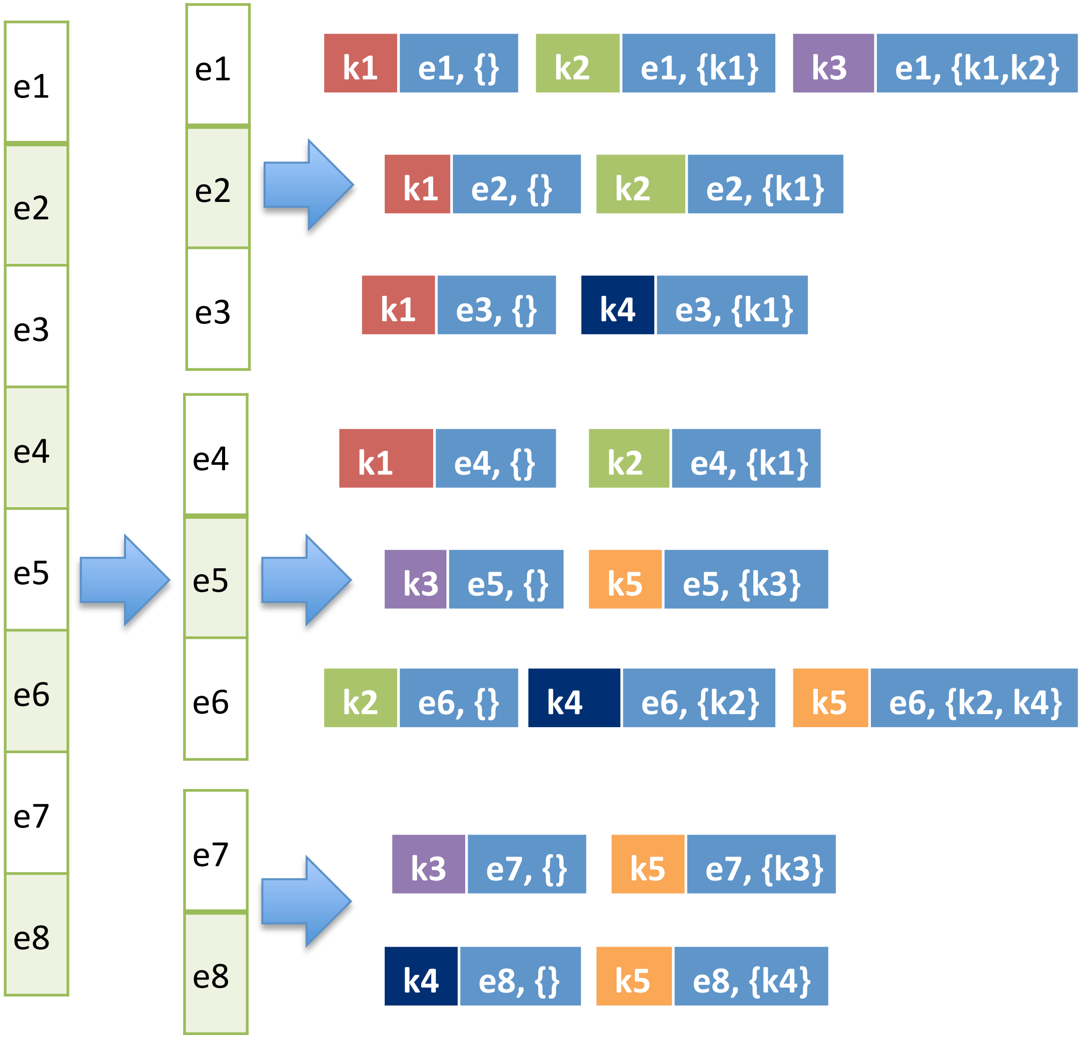
Adopt Comparison Propagation [Papadakis et al. 2012] to MapReduce:

- 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



e1
e2
e3
e4
e5
e6
e7
e8

e1
e2
e3

k1 e1, {} k2 e1, {k1} k3 e1, {k1,k2}

k1 e2, {} k2 e2, {k1}

k1 e3, {} k4 e3, {k1}

e4
e5
e6

k1 e4, {} k2 e4, {k1}

k3 e5, {} k5 e5, {k3}

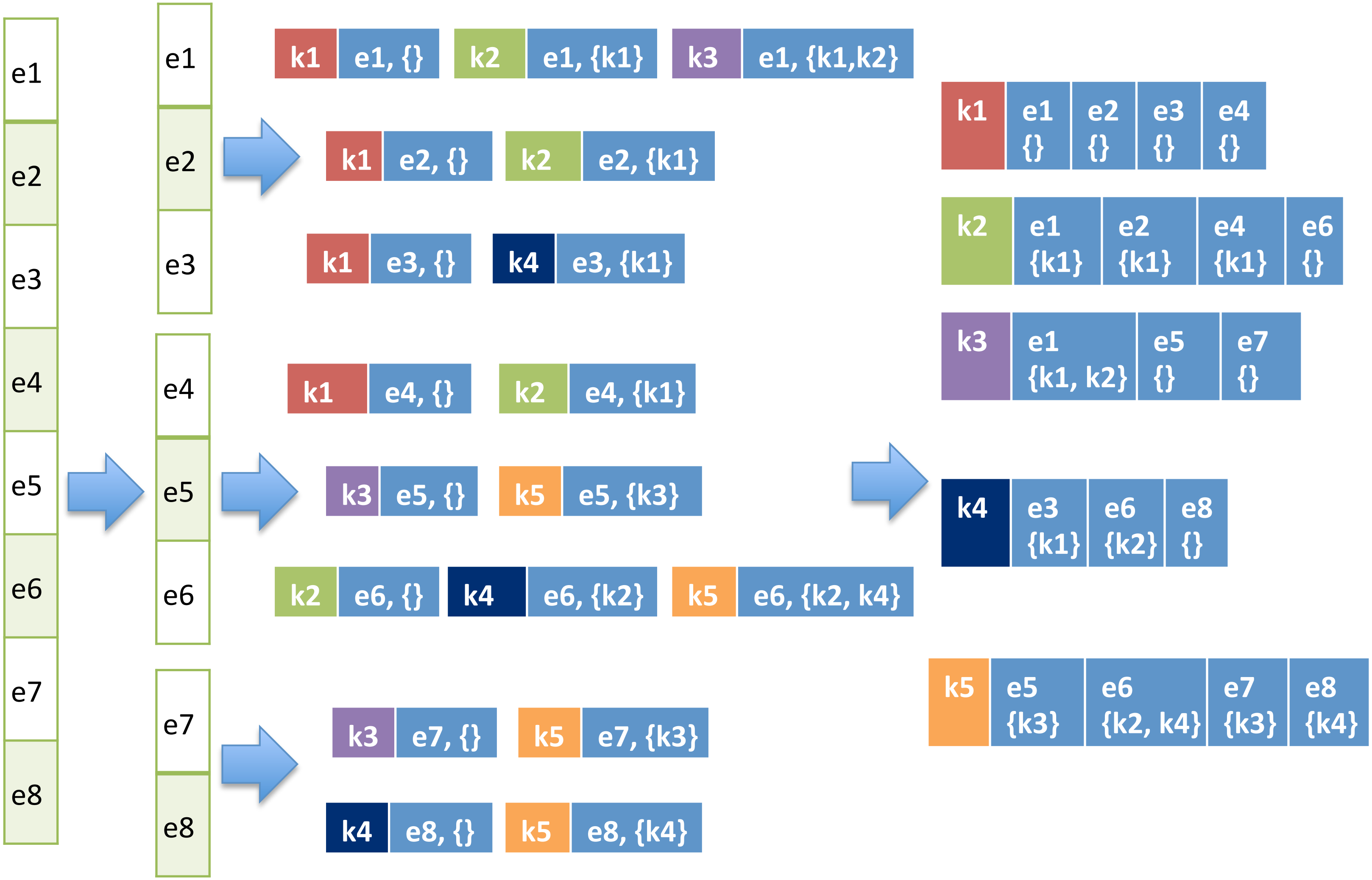
k2 e6, {} k4 e6, {k2} k5 e6, {k2, k4}

e7
e8

k3 e7, {} k5 e7, {k3}

k4 e8, {} k5 e8, {k4}

Map: Append the Subset of Smaller Keys for the Same Description



e1
e2
e3
e4
e5
e6
e7
e8

e1
e2
e3

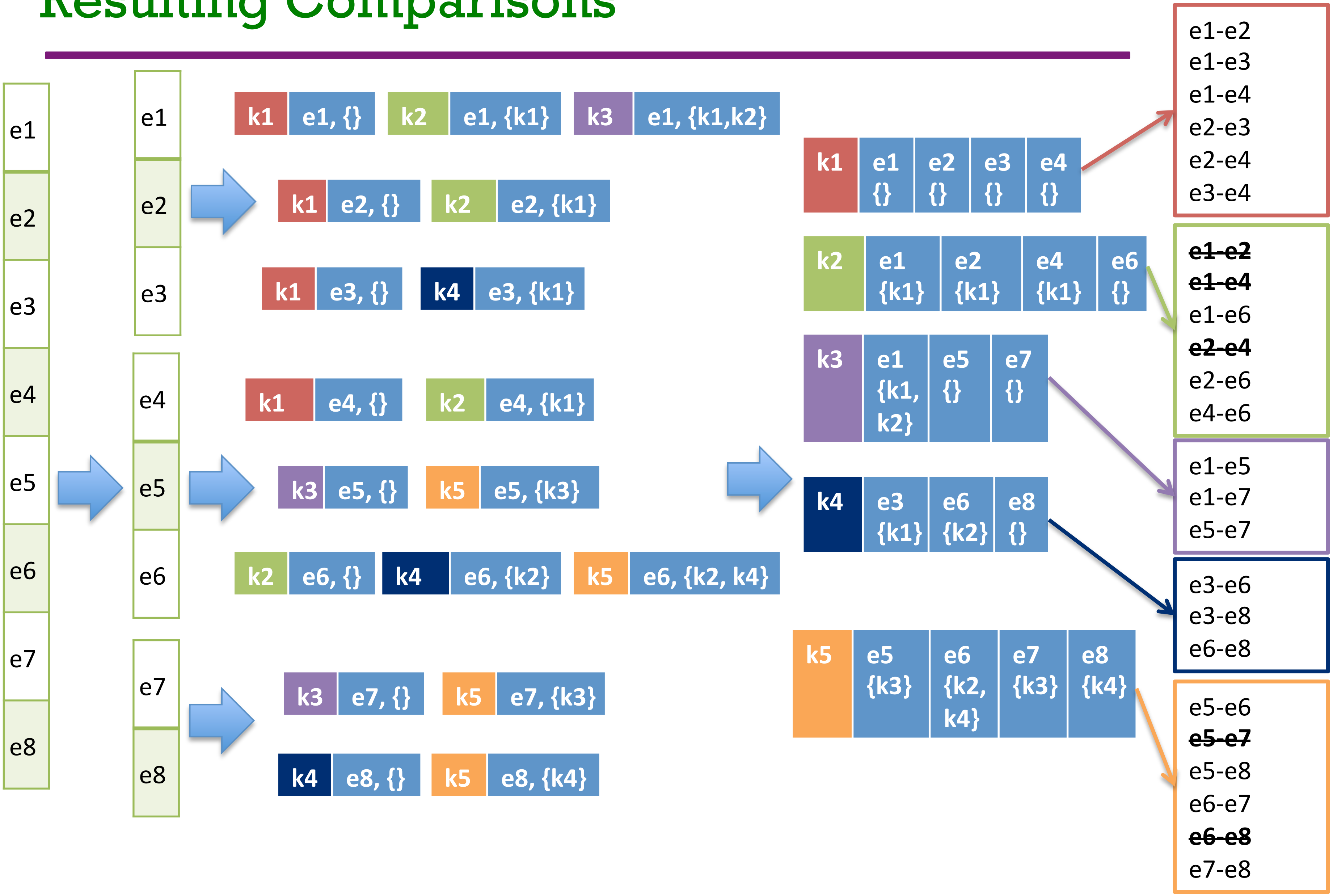
e4
e5
e6

e7
e8

k1 e1, {} k2 e1, {k1} k3 e1, {k1,k2}
k1 e2, {} k2 e2, {k1}
k1 e3, {} k4 e3, {k1}
k1 e4, {} k2 e4, {k1}
k3 e5, {} k5 e5, {k3}
k2 e6, {} k4 e6, {k2} k5 e6, {k2, k4}
k3 e7, {} k5 e7, {k3}
k4 e8, {} k5 e8, {k4}

k1 e1 {} e2 {} e3 {} e4 {}
k2 e1 {k1} e2 {k1} e4 {k1} e6 {}
k3 e1 {k1, k2} e5 {} e7 {}
k4 e3 {k1} e6 {k2} e8 {}
k5 e5 {k3} e6 {k2, k4} e7 {k3} e8 {k4}

Resulting Comparisons



e1
e2
e3
e4
e5
e6
e7
e8

e1
e2
e3

k1 e1, {} k2 e1, {k1} k3 e1, {k1,k2}
k1 e2, {} k2 e2, {k1}
k1 e3, {} k4 e3, {k1}

e4
e5
e6

k1 e4, {} k2 e4, {k1}
k3 e5, {} k5 e5, {k3}
k2 e6, {} k4 e6, {k2} k5 e6, {k2, k4}

e7
e8

k3 e7, {} k5 e7, {k3}
k4 e8, {} k5 e8, {k4}

k1 e1 {} e2 {} e3 {} e4 {}

k2 e1 {k1} e2 {k1} e4 {k1} e6 {}

k3 e1 {k1, k2} e5 {} e7 {}

k4 e3 {k1} e6 {k2} e8 {}

k5 e5 {k3} e6 {k2, k4} e7 {k3} e8 {k4}

e1-e2
e1-e3
e1-e4
e2-e3
e2-e4
e3-e4

~~e1-e2~~
~~e1-e4~~
e1-e6
~~e2-e4~~
e2-e6
e4-e6

e1-e5
e1-e7
e5-e7

e3-e6
e3-e8
e6-e8

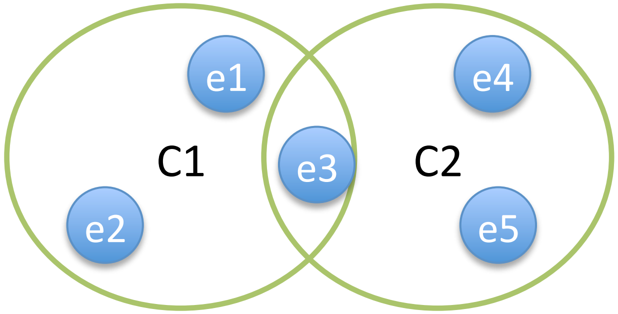
e5-e6
~~e5-e7~~
e5-e8
e6-e7
~~e6-e8~~
e7-e8

Large-Scale Collective Entity Matching
[Rastogi et al. 2011]

Assume that there is a rule *R: Match(e1, e2) => Match(e4, e5)*

and that we have inferred: *Match(e1, e2)*

In C2, we cannot infer *Match(e4, e5)*



map: assign each Ci to a cluster node and run entity resolution on it

reduce: bring all the new evidence for each Ci together

We should somehow inform C2 that e1 matches e2

- Then we could infer that e4 matches e5, according to rule R

Solution: message passing

- After matching in C1 finishes, send a message "Match(e1, e2)"
- In the next MapReduce round, entity resolution runs with the new evidence and infers *Match(e4, e5)*

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)

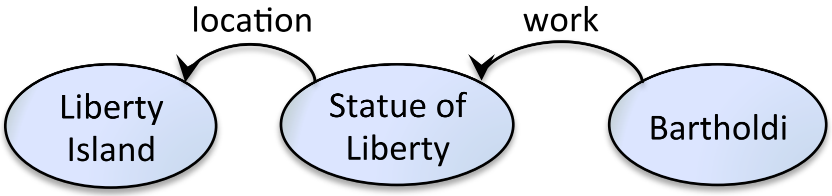
- Matches are identified using two kinds of similarities:
  - String similarity (token-based) of their literal values L(e)
    - Checked once
  - Contextual similarity (based on neighbors in the entity graph)
    - Checked iteratively

Contextual Similarity

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(Statue of liberty) =
{(location, Liberty Island, w1),
(*is* work *of*, Bartholdi, w2)}

Contextual Similarity

The contextual similarity of nodes n and m is:

context_sim(n, m) =

- $\sum_{(p_i, z_i, w_i) \in C(n)} \max_{(p_j, z_j, w_j) \in C(m)} w_i \cdot x_{z_i, z_j} \cdot sim(p_i, p_j), if\ |C(n)| \le |C(m)|$
- $\sum_{(p_j, z_j, w_j) \in C(m)} \max_{(p_i, z_i, w_i) \in C(n)} w_j \cdot x_{z_i, z_j} \cdot sim(p_i, p_j), else$

where

$x_{n,m}$ is 1, if n, m are identified as matches, and 0, else

$sim(p_i, p_j)$ 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:

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

β controls the contextual influence

θ 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

LINDA – Distributed Entity Resolution Using MapReduce

Distribute across a cluster the input entity graph

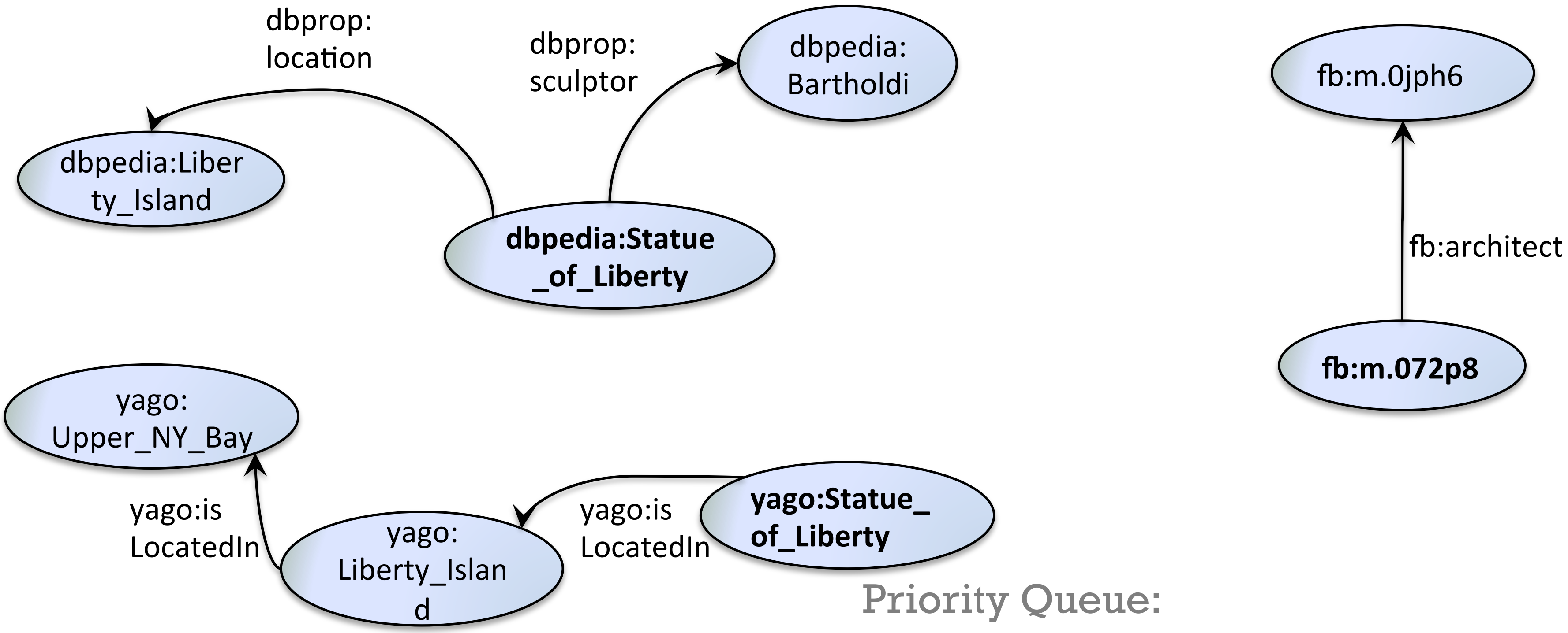
- A node i holds a portion Q_i of the priority queue and the respective part G_i of the graph

Map phase

- Mapper i reads Q_i and forwards messages to reducers for similarities re-computations
  - 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)

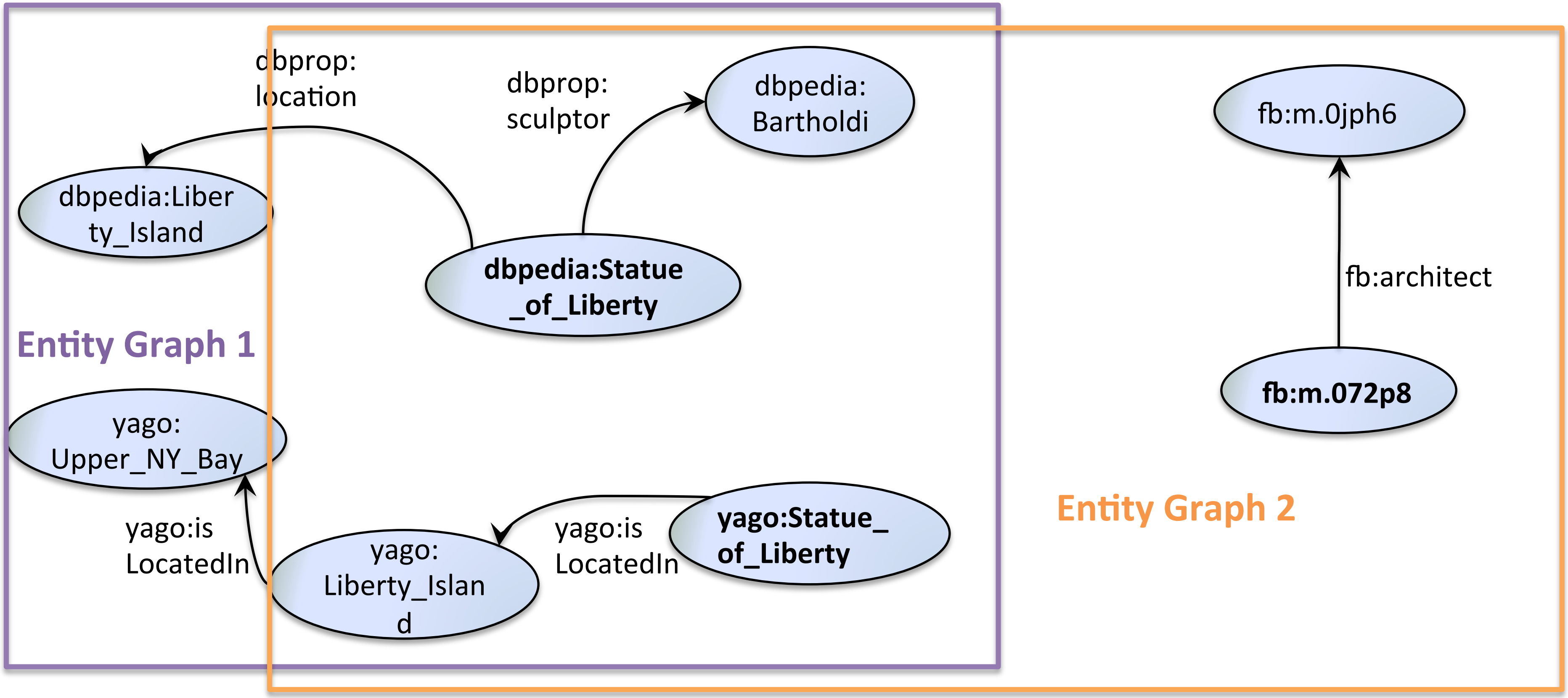
Priority Queue 1 (machine 1):

(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



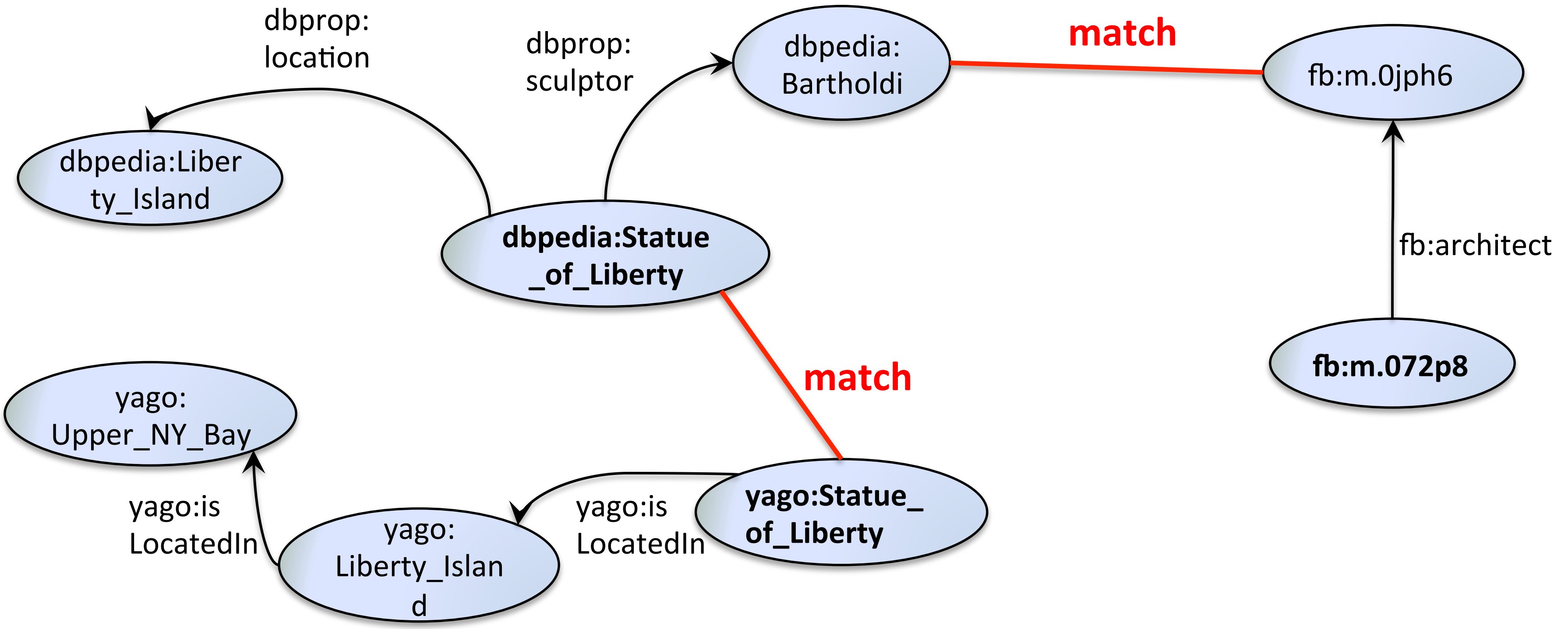
Priority Queue 1 (machine 1):

(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



Priority Queue 1:

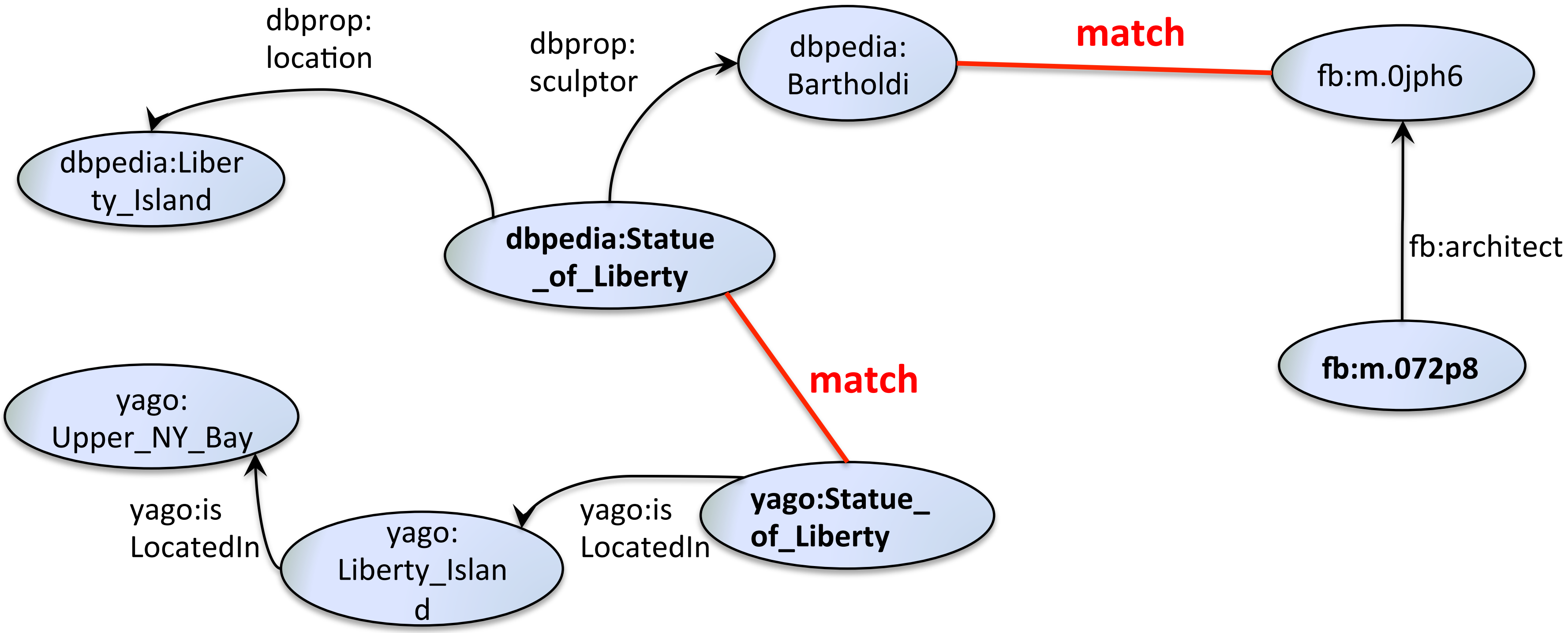
(dbpedia:Statue_of_Liberty, yago:Statue_of_Liberty)
(dbpedia:Statue_of_Liberty, yago:Liberty_Island)
(dbpedia:Liberty_Island,yago:Upper_NY_Bay)
(dbpedia:Liberty_Island, yago:Liberty_Island)
(dbpedia:Liberty_Island, yago:Statue_of_Liberty)

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



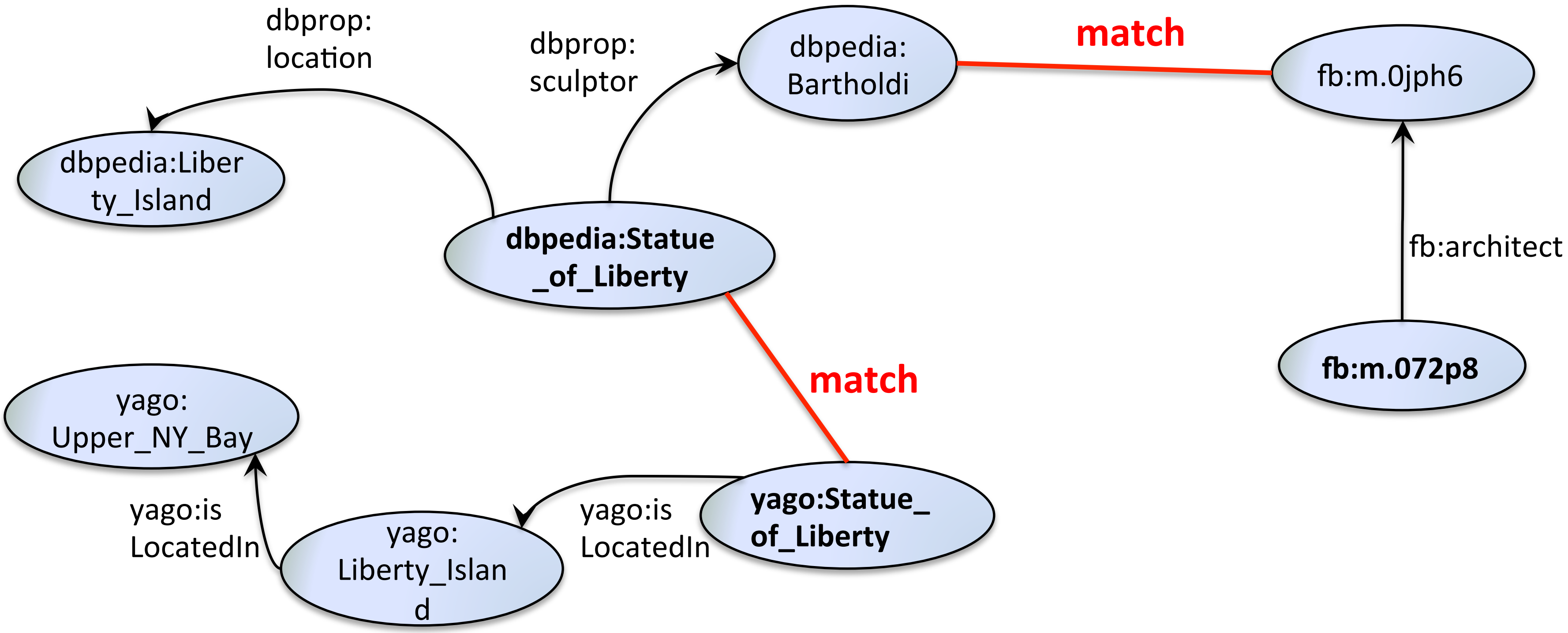
Priority Queue 1:

(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)~~

Dequeue these pairs, as each entity can be mapped to at most one entity per data source



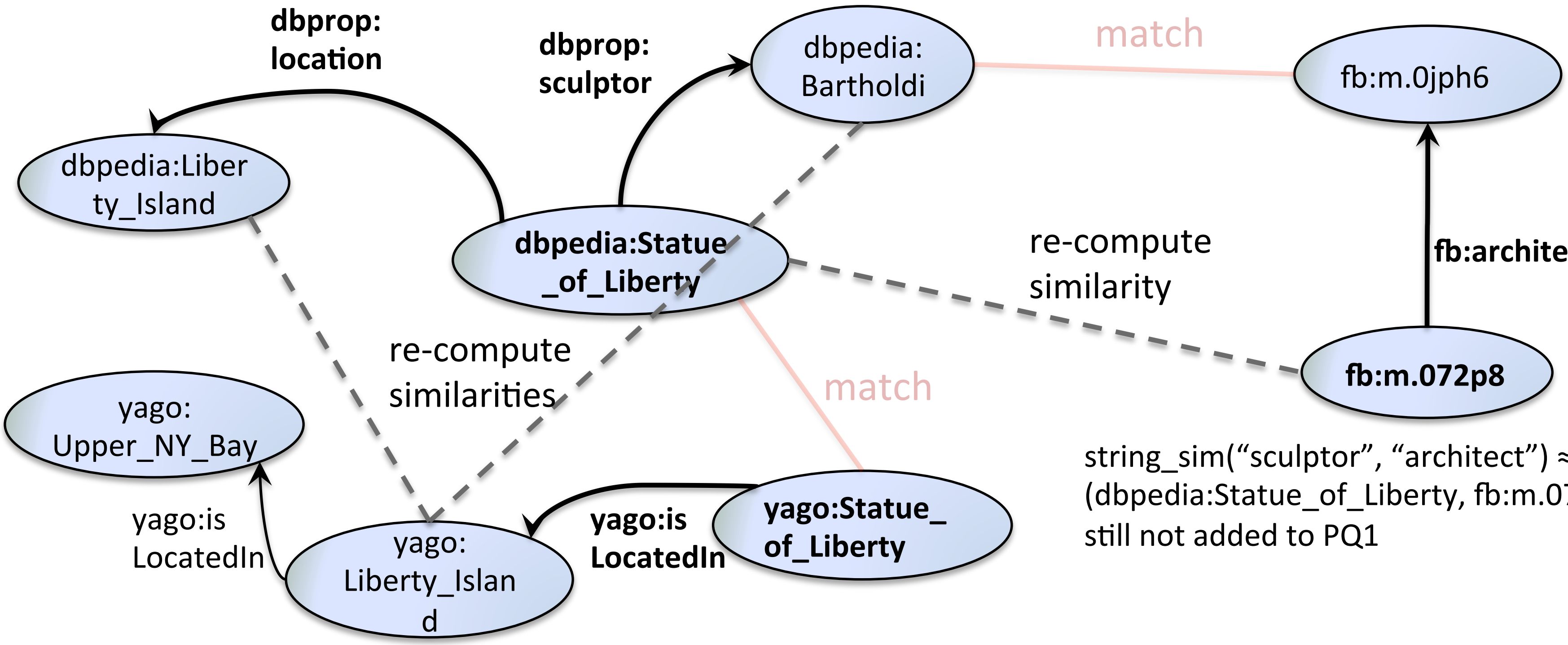
Priority Queue 1:

(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



string_sim("sculptor", "architect") ≈ 0, so (dbpedia:Statue_of_Liberty, fb:m.072p8) is still not added to PQ1

Priority Queue 1:

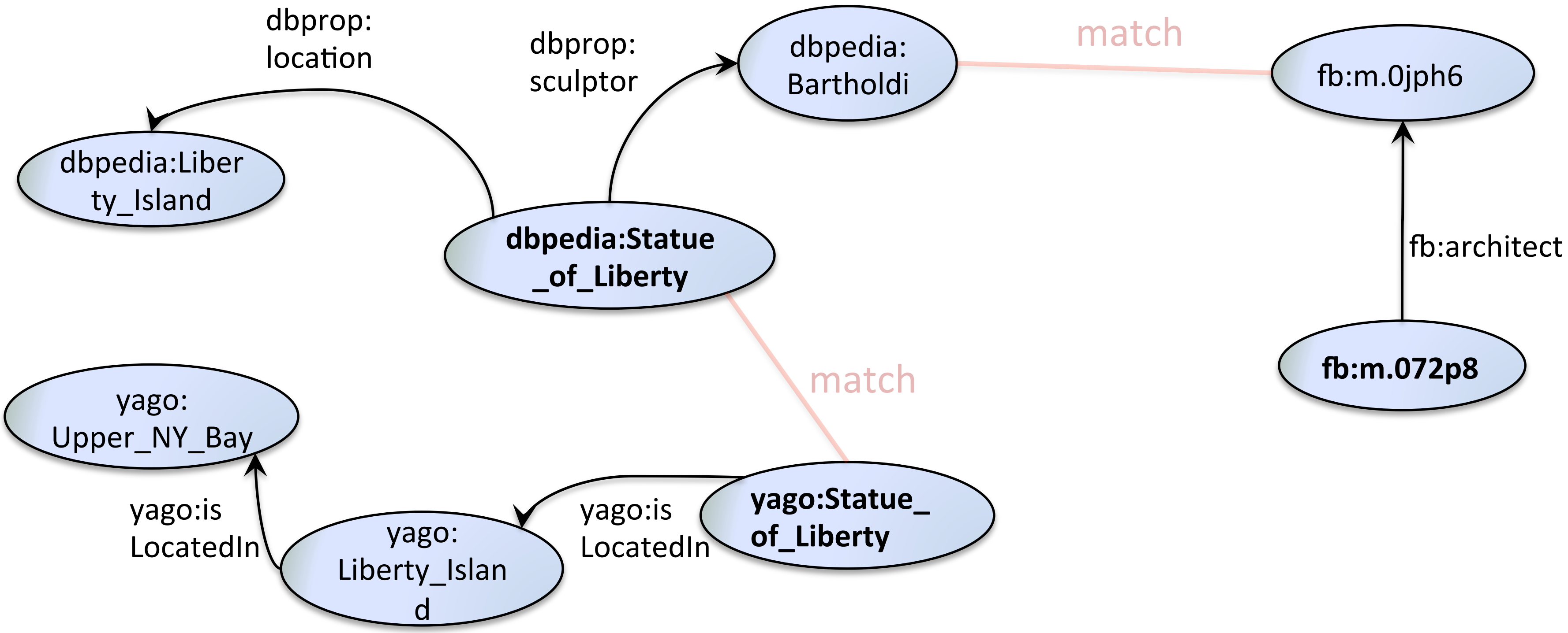
(dbpedia:Liberty_Island,yago:Upper_NY_Bay)

(dbpedia:Liberty_Island, yago:Liberty_Island)

Priority Queue 2:

Contextual similarity re-computations

Property names are also taken into account

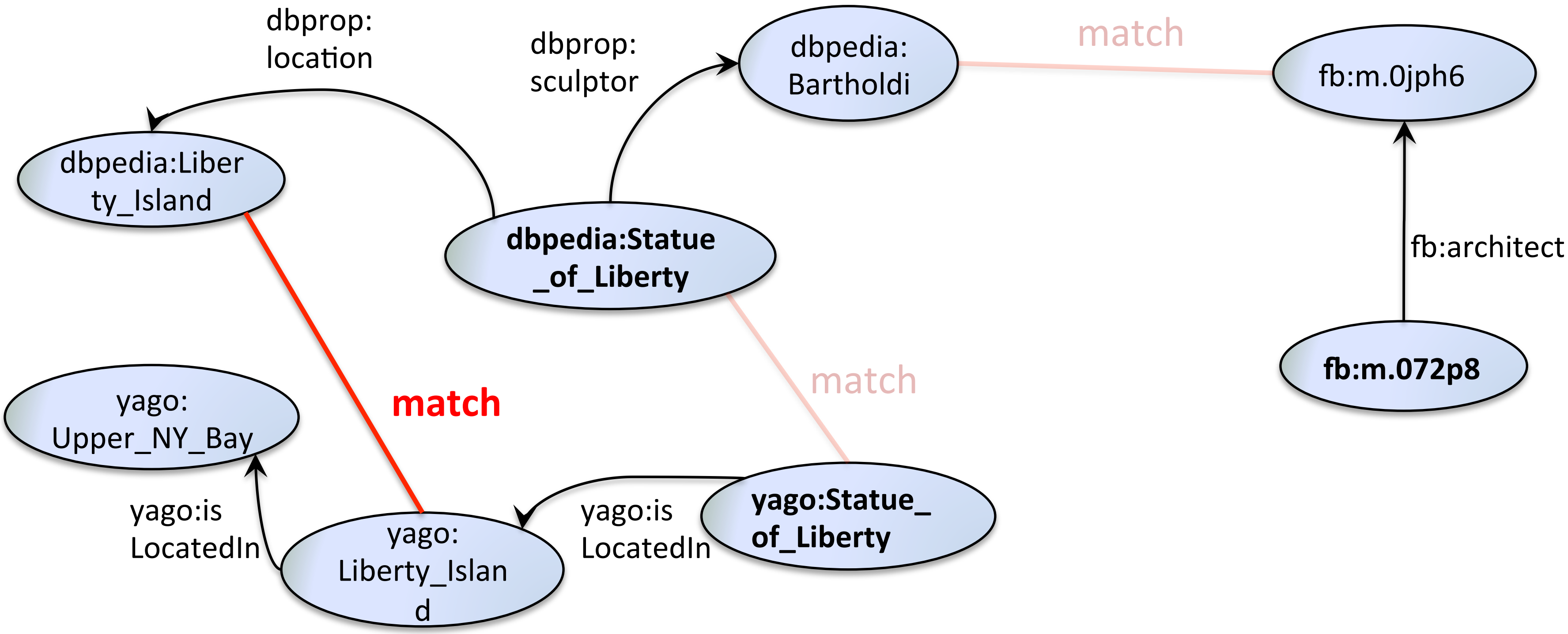


Priority Queue 1:

(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:

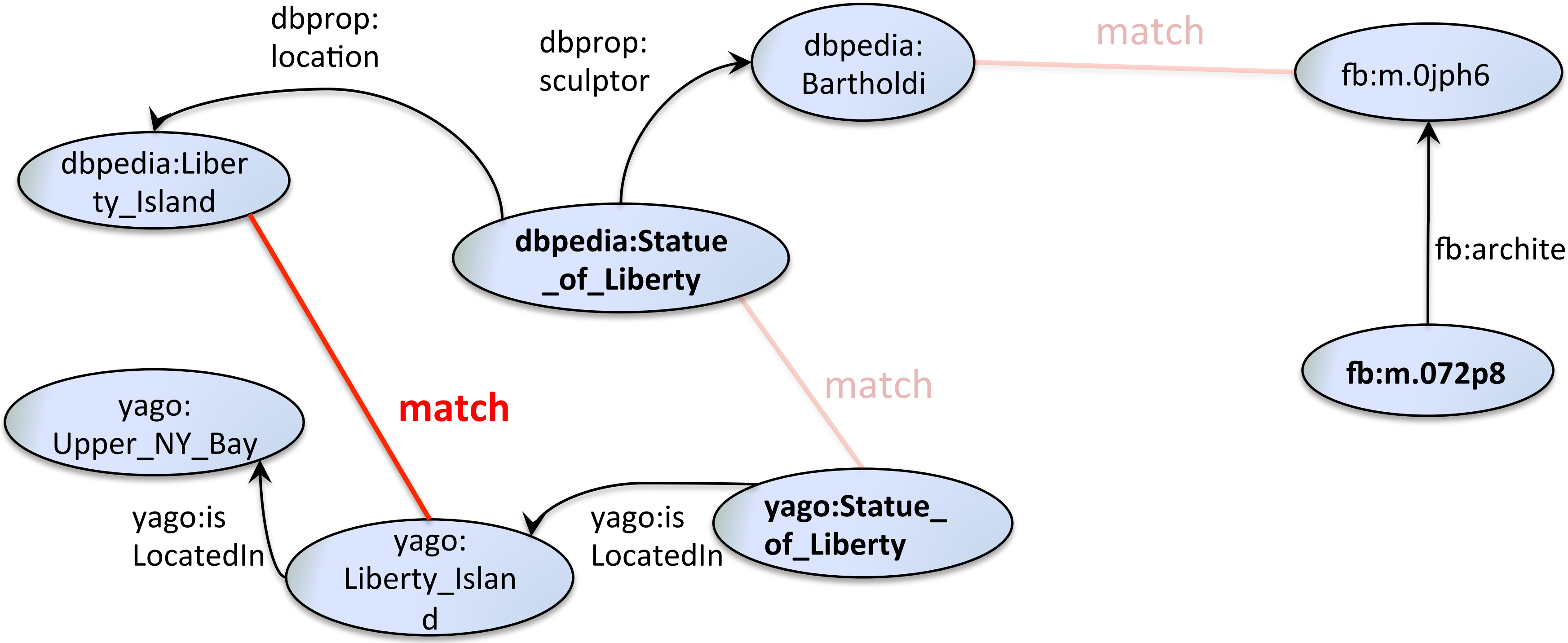
(dbpedia:Liberty_Island, yago:Liberty_Island)

(dbpedia:Liberty_Island,yago:Upper_NY_Bay)

Priority Queue 2:

The head of each queue is a match by default

This triggers update messages



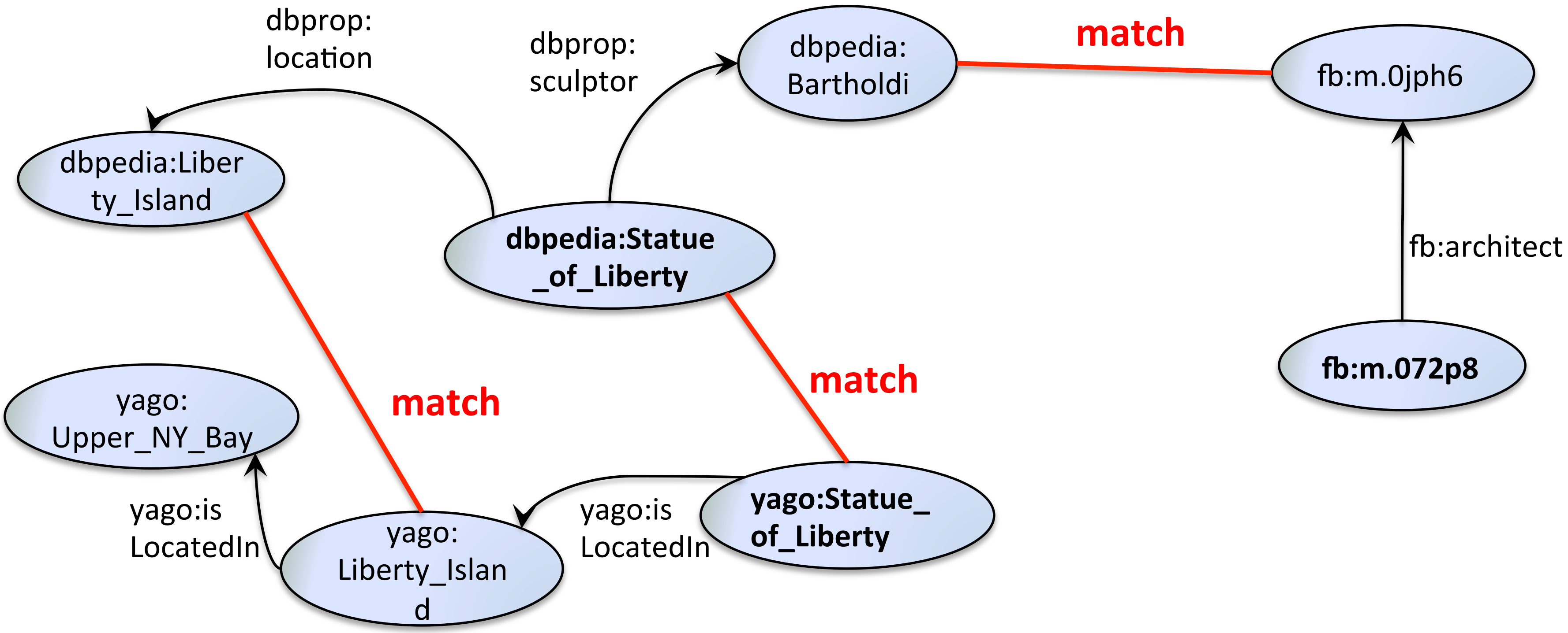
Priority Queue 1:

(dbpedia:Liberty_Island, yago:Liberty_Island)

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

Priority Queue 2:

Dequeue this pair, as each entity can be mapped to at most one entity per data source

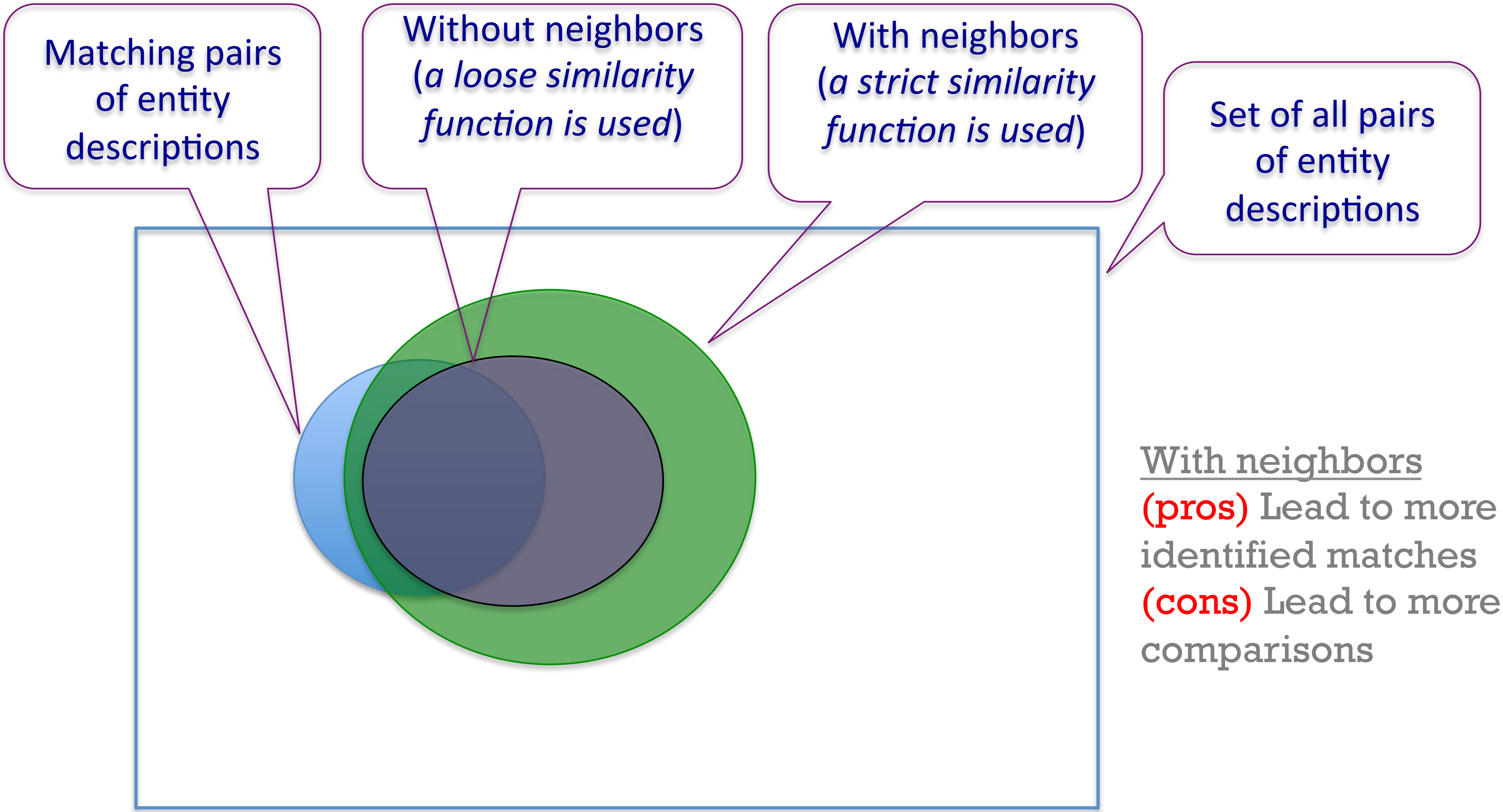


Priority Queue 1:

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

Measures for Effectiveness

- **Precision**: number of correctly identified matches, compared to the number of all suggested matches (correctly or incorrectly)

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

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

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

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

$$\text{F} - \text{score} = 2\frac{\Pr ecision \cdot \text{Re}\,call}{\Pr ecision + \text{Re}\,call}$$

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

GMD Example – 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 (merge): (e1), (e2) → (e1, e2) Cost:**1**

GMD(X,Y) = 1

GMD Example – 2

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): {(e1, e2, e3), (e4)} → {(e1, e2), (e3), (e4)} Cost: **2**

Transformation (merge): {(e1, e2), (e3), (e4)} → {(e1, e2), (e3, e4)} Cost: **1**

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

[Papadakis et al. 2012]

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

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 *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
  - 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

match(e_1, e_2) => match (e_2,e_1)

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

Measures for Efficiency

Comparison cardinality (CC) approximates the reduction ratio

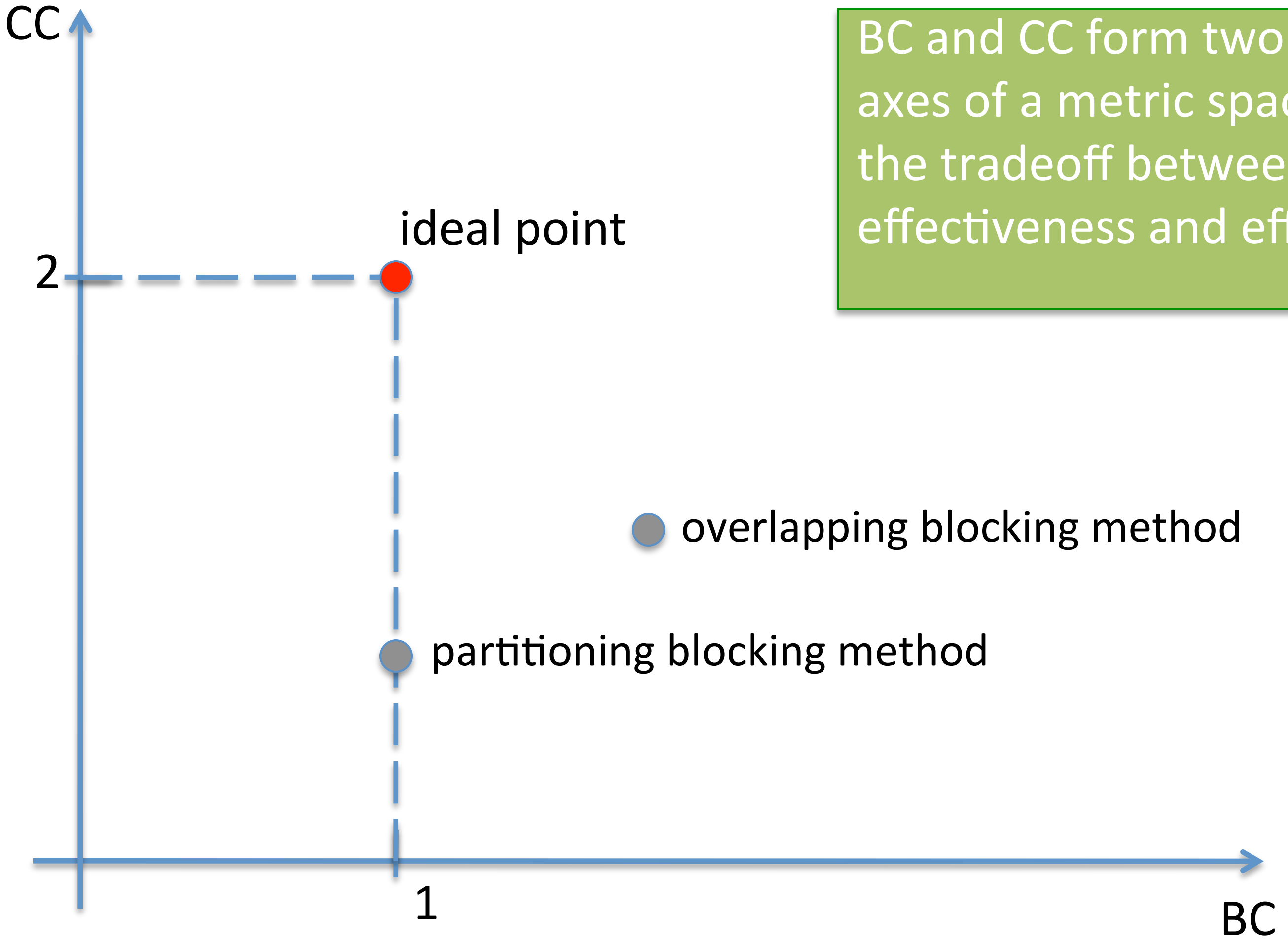
- CC is the average number of block assignments per comparison

[Papadakis et al. 2012]

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

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

Papadakis et al. 2013(a)

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

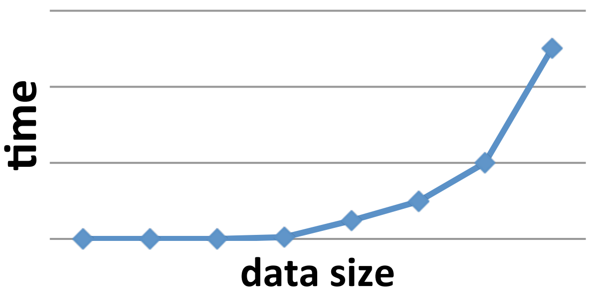
Scalable methods can handle entity resolution in large volumes of data, namely in the scale of millions or billions of entity descriptions

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

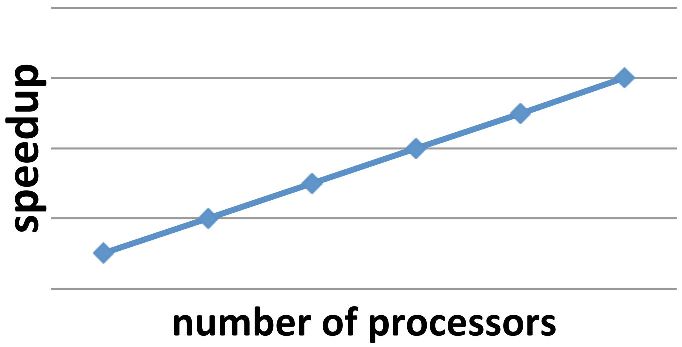
[used in distributed computing]

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

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 and e_j 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. $x_{ei,ej}$ = [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 and e_j 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

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

[Fellegi & Sunter 1969]

q = (e_i, e_j), γ is the comparison vector of e_i, e_j

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

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)

$$R > t \Rightarrow q \in M$$
$$t' \leq R \leq t \Rightarrow q \in A$$
$$R < t' \Rightarrow q \in Q$$

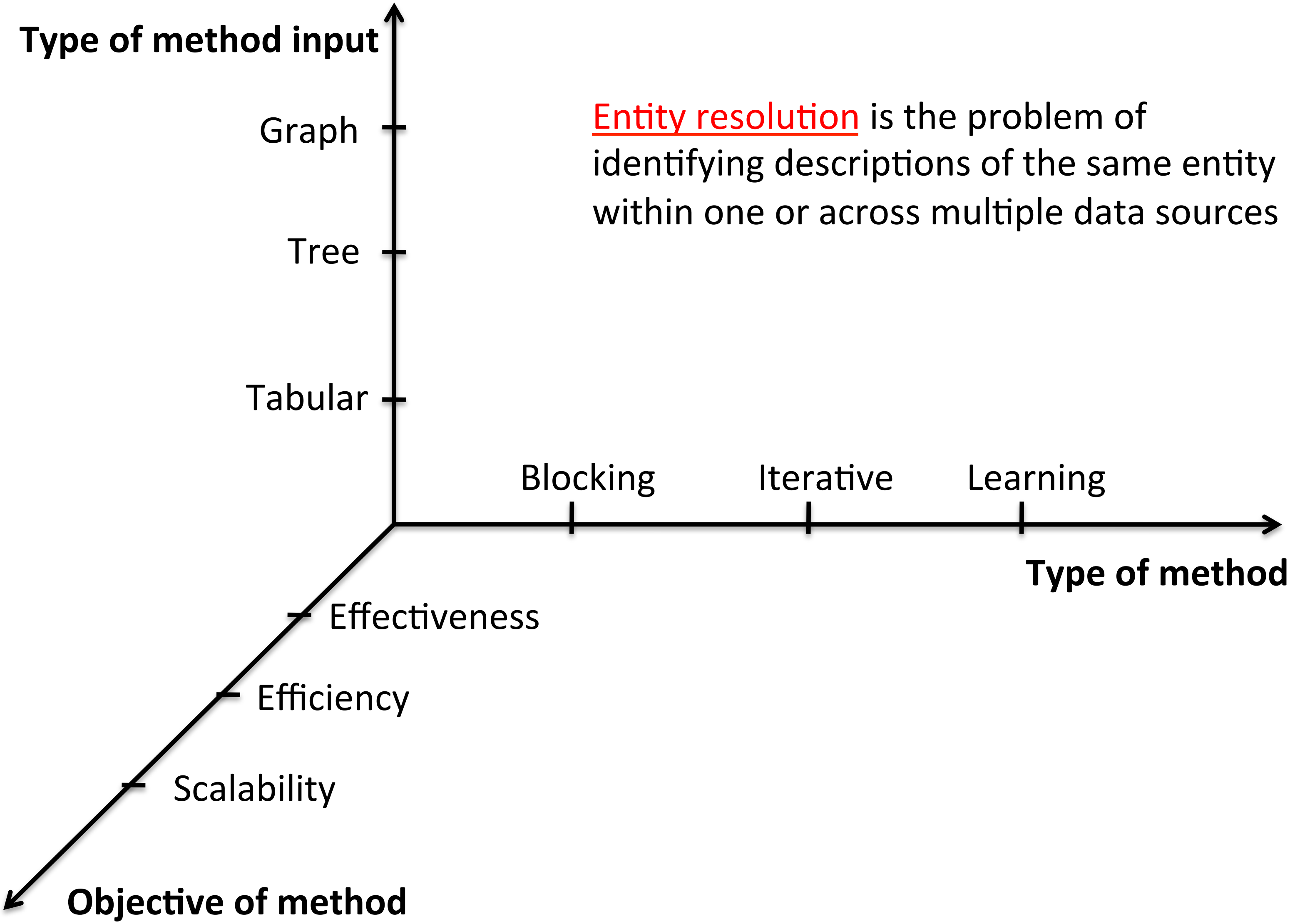
In brief, existing approaches use:

- *Supervised learning techniques, active learning techniques, unsupervised learning techniques*

Conclusions

Solution Space

Entity resolution is the problem of identifying descriptions of the same entity within one or across multiple data sources



Type of method input

Graph

Tree

Tabular

Blocking

Iterative

Learning

Type of method

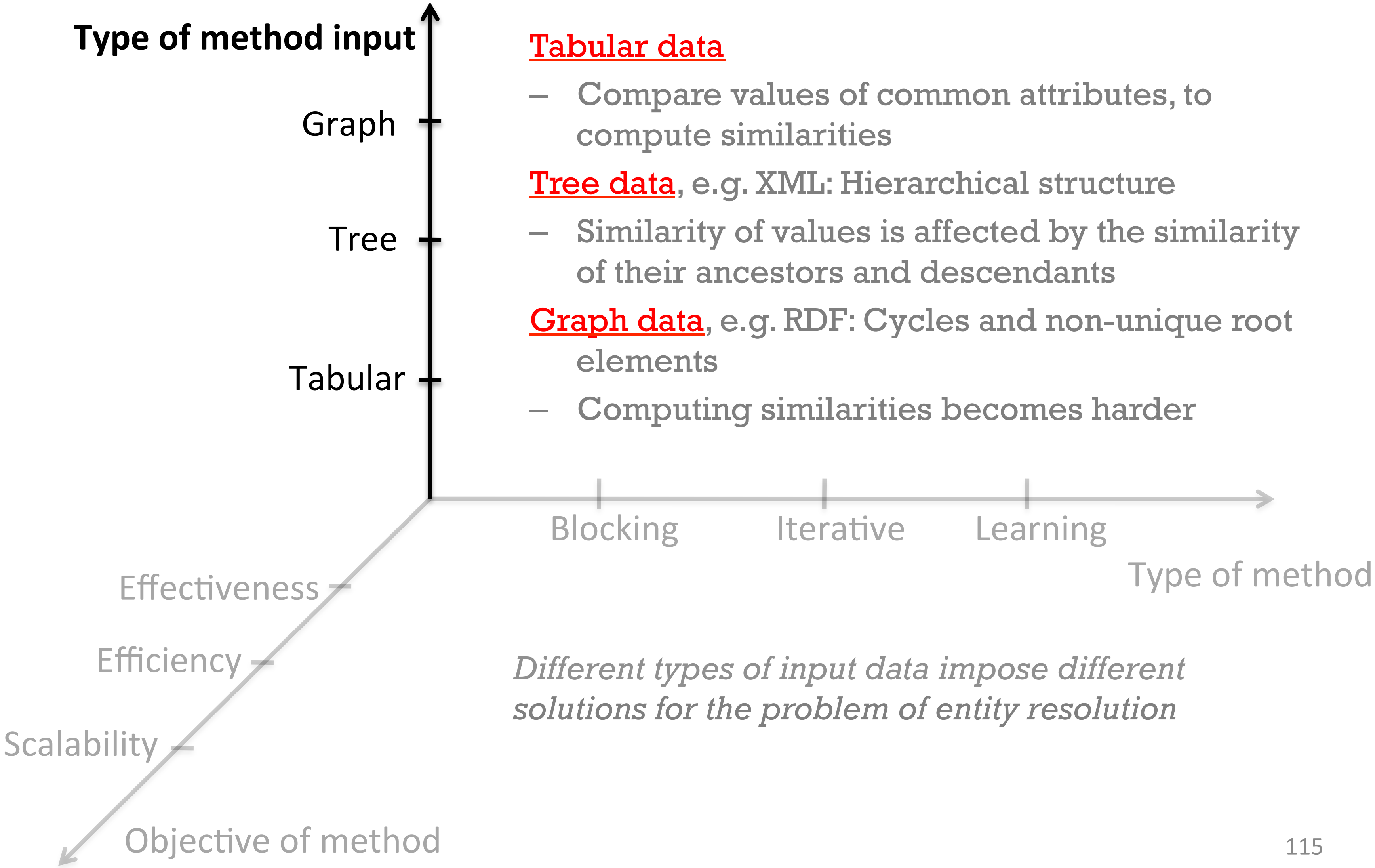
Effectiveness

Efficiency

Scalability

Objective of method

Solution Space



Tabular data

- Compare values of common attributes, to compute similarities

Tree data, e.g. XML: Hierarchical structure

- Similarity of values is affected by the similarity of their ancestors and descendants

Graph data, e.g. RDF: Cycles and non-unique root elements

- Computing similarities becomes harder

Different types of input data impose different solutions for the problem of entity resolution

Solution Space

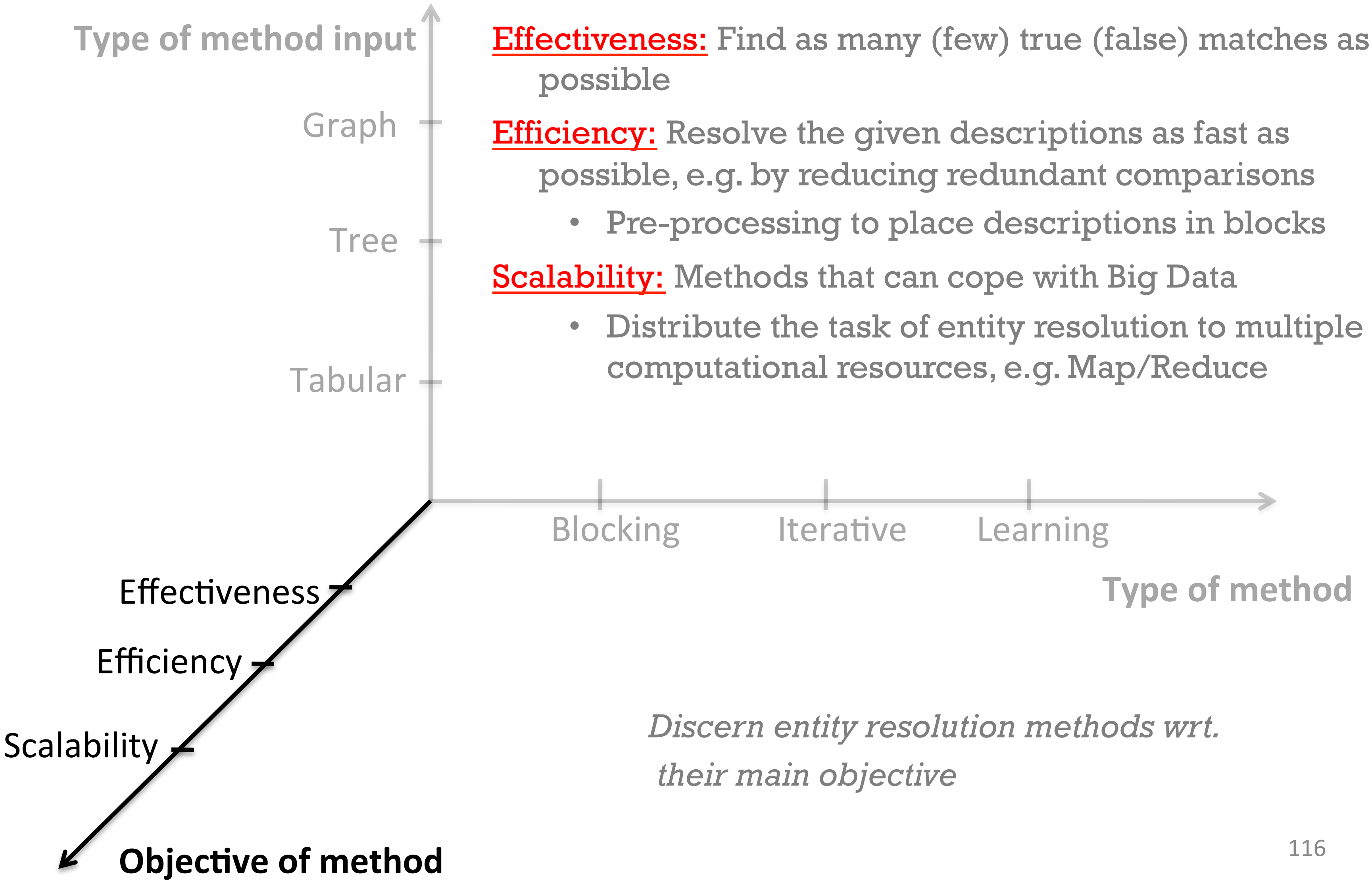
Effectiveness: Find as many (few) true (false) matches as possible

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

Blocking: Group together descriptions close to each other

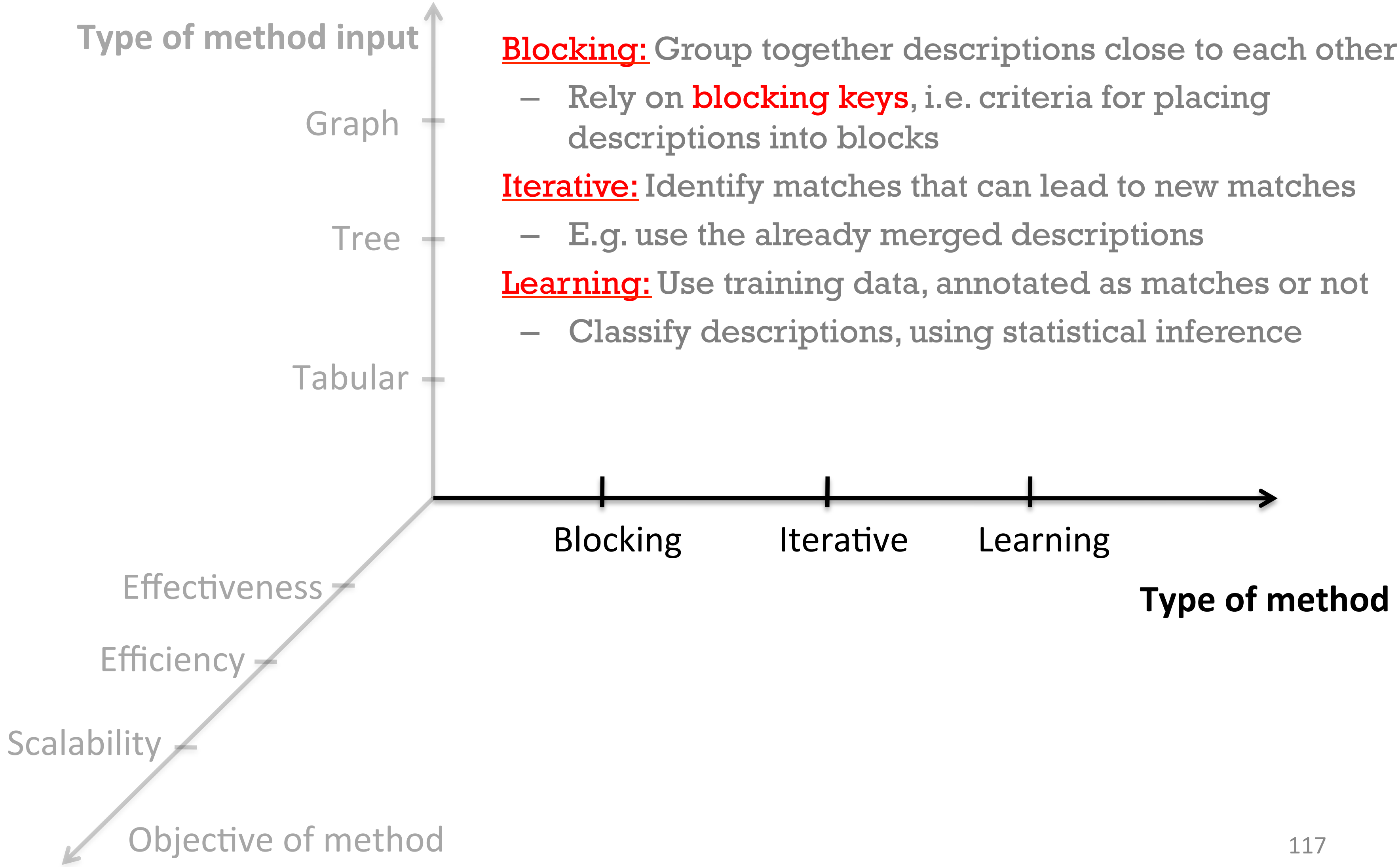
- Rely on **blocking keys**, i.e. criteria for placing descriptions into blocks

Iterative: Identify matches that can lead to new matches

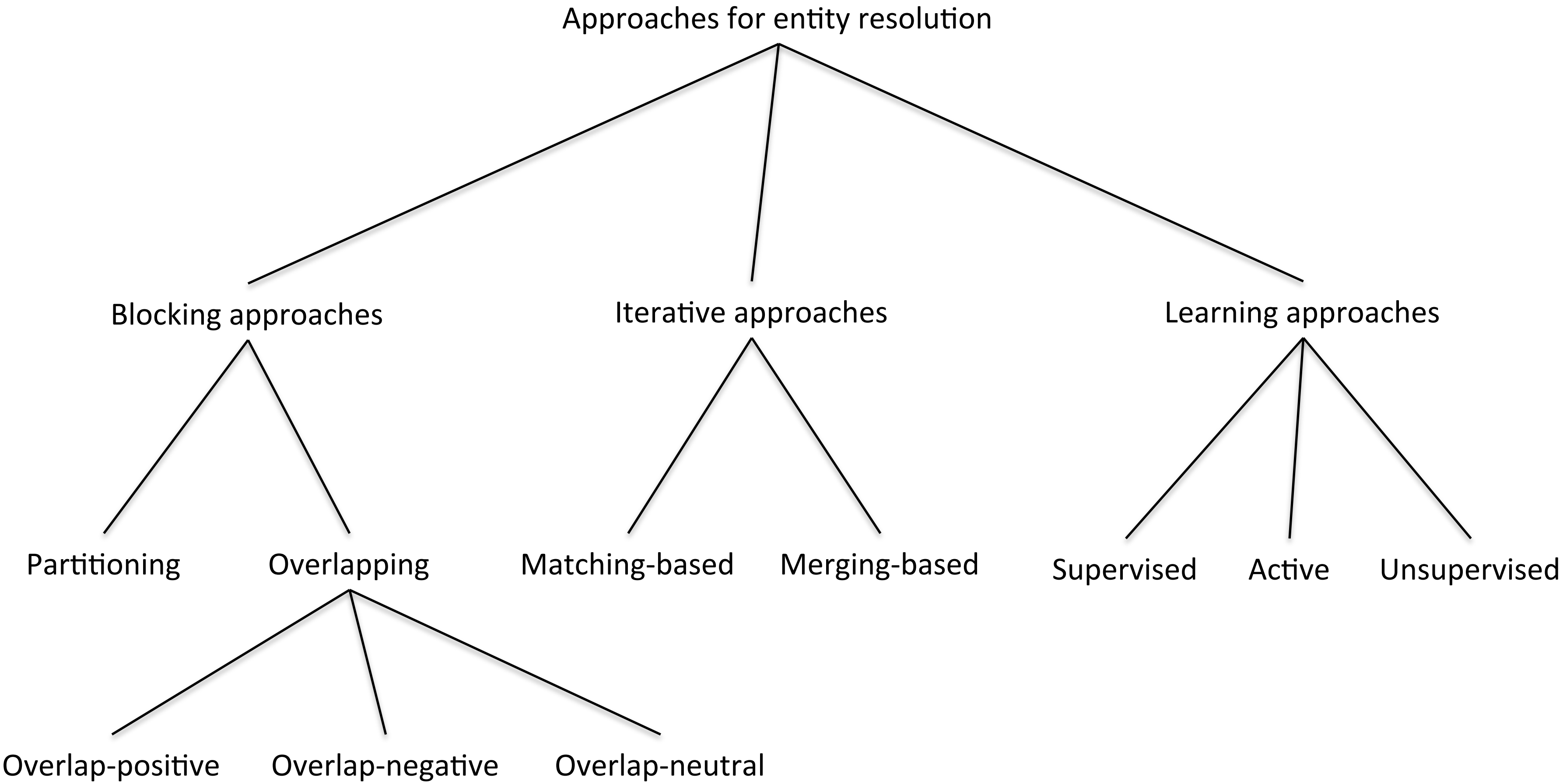
- E.g. use the already merged descriptions

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

Iterative approaches

Learning approaches

Partitioning

Overlapping

Matching-based

Merging-based

Supervised

Active

Unsupervised

Overlap-positive

Overlap-negative

Overlap-neutral

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 → very similar
- Overlap-negative: few common blocks → 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

Approach	Partitioning	Overlapping		
		positive	negative	neutral
Fellegi & Sunter 1969	•			
Hernandez & Stolfo 1995				•
Yan et al. 2007	•			
Draisbach & Naumann 2009				•
McCallum et al. 2000			•	
Christen 2012			•	
Gravano et al. 2001		•		
Aizawa & Oyama 2005		•		
Jin et al. 2003		•		
Kolb et al. 2011, 2012	•			
Papadakis et al. 2011		+		
Papadakis et al. 2013 (a)		+		
Papadakis et al. 2013 (b)		+		
Papadakis et al. 2012		+		

•: 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

Approach	Matching-based	Merging-based
Bhattacharya & Getoor 2004, 2007	•	
Rastogi et al. 2011	•	
Dong et al. 2005	•	
Herschel et al. 2012	•	
Weis & Naumann 2006	☐	
Weis & Naumann 2004	☐	
Leitão et al. 2007, 2013	☐	
Puhlmann et al. 2006	☐	
Böhm et al. 2012	+	
Benjelloun et al. 2009		•
Benjelloun et al. 2007		•
Whang et al. 2009		•
Kim & Lee 2010		•

• : 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

Type of method input	Approach	Blocking	Iterative	Learning
Tabular	Fellegi & Sunter 1969	☐		
	Hernandez & Stolfo 1995	☐		
	Yan et al. 2007	☐		
	Draisbach & Naumann 2009	☐		
	McCallum et al. 2000	☐		
	Christen 2012	☐		
	Gravano et al. 2001	☐		
	Aizawa & Oyama 2005	☐		
	Jin et al. 2003	☐		
	Kolb et al. 2011, 2012	☐ +		
	Benjelloun et al. 2009		☐	
	Benjelloun et al. 2007		+	
	Whang et al. 2009	• ☐ +	• ☐ +	
	Kim & Lee 2010	• ☐ +	• ☐ +	
	Herschel et al. 2012		• +	
	Dong et al. 2005		•	
	Bhattacharya & Getoor 2004, 2007		•	
	Rastogi et al. 2011		+	
	Cochinwala et al. 2001			• ☐
	Bilenko & Mooney 2003			•
	Christen 2008			•
	Chen et al. 2009			• ☐
	Ravikumar & Cohen 2004			•
	Bhattacharya & Getoor 2006			•
	Sarawagi & Bhamidipaty 2002	• ☐		• ☐
	Tejada et al. 2002, 2001			•
	Wang et al. 2012			•
	Verykios et al. 2000			•
Tree	Weis & Naumann 2006		•	
	Weis & Naumann 2004		•	
	Leitao et al. 2007, 2013		•	
	Puhlmann et al. 2006		•	
Graph	Papadakis et al. 2011	• ☐		
	Papadakis et al. 2013 (a)	• ☐		
	Papadakis et al. 2013 (b)	☐ +		
	Papadakis et al. 2012	☐ +		
	Bohm et al. 2012		• +	

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, ect.
- 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

- Menestrina, D., Whang, S., Garcia-Molina, H.: Evaluating entity resolution results. PVLDB 3(1), 208–219 (2010)
- Papadakis, G., Ioannou, E., Niederee, C., Palpanas, T., Nejdl, W.: Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data. In: WSDM, pp. 53–62 (2012)
- Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of the American Statistical Association 64(328), 1183–1210 (1969)
- Hernandez, M.A., Stolfo, S.J.: The merge/purge problem for large databases. In: SIGMOD (1995)
- Yan, S., Lee, D., Kan, M.Y., Giles, C.L.: Adaptive sorted neighborhood methods for efficient record linkage. In: JCDL, pp. 185–194 (2007)
- Draisbach, U., Naumann, F.: A comparison and generalization of blocking and windowing algorithms for duplicate detection. In: QDB, pp. 51–56 (2009)
- McCallum, A., Nigam, K., Ungar, L.H.: Efficient clustering of high-dimensional data sets with application to reference matching. In: KDD, pp. 169–178 (2000)
- Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. IEEE Trans. Knowl. Data Eng. 24(9), 1537–1555 (2012)
- Gravano, L., Ipeirotis, P.G., Jagadish, H.V., Koudas, N., Muthukrishnan, S., Srivastava, D.: Approximate string joins in a database (almost) for free. In: VLDB, pp. 491– 500 (2001)
- Aizawa, A.N., Oyama, K.: A fast linkage detection scheme for multi-source information integration. In: WIRI, pp. 30–39 (2005)
- Jin, L., Li, C., Mehrotra, S.: Efficient record linkage in large data sets. In: DASFAA, pp. 137–146 (2003)
- Draisbach, U., Naumann, F., Szott, S., Wonneberg, O.: Adaptive Windows for Duplicate Detection. ICDE 2012: 1073-1083

References

- Kolb, L., Thor, A., Rahm, E.: Block-based Load Balancing for Entity Resolution with MapReduce. In: CIKM, pp. 2397–2400 (2011)
- Kolb, L., Thor, A., Rahm, E.: Dedoop: Efficient Deduplication with Hadoop. PVLDB 5(12), 1878–1881 (2012)
- Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S.E., Widom, J.: Swoosh: a generic approach to entity resolution. VLDB J. 18(1), 255–276 (2009)
- Benjelloun, O., Garcia-Molina, H., Gong, H., Kawai, H., Larson, T.E., Menestrina, D., Thavisomboon, S.: D-swoosh: A family of algorithms for generic, distributed entity resolution. In: ICDCS, p. 37 (2007)
- Whang, S.E., Menestrina, D., Koutrika, G., Theobald, M., Garcia-Molina, H.: Entity resolution with iterative blocking. In: SIGMOD, pp. 219–232 (2009)
- Kim, H., Lee, D.: Harra: fast iterative hashed record linkage for large-scale data collections. In: EDBT, pp. 525– 536 (2010)
- Herschel, M., Naumann, F., Szott, S., Taubert, M.: Scalable iterative graph duplicate detection. IEEE Trans. Knowl. Data Eng. 24(11), 2094–2108 (2012)
- Dong, X., Halevy, A.Y., Madhavan, J.: Reference reconciliation in complex information spaces. In: SIGMOD, pp. 85–96 (2005)
- Bhattacharya, I., Getoor, L.: Iterative record linkage for cleaning and integration. In: DMKD (2004)
- Bhattacharya, I., Getoor, L.: Collective entity resolution in relational data. TKDD 1(1) (2007)
- Cochinwala, M., Kurien, V., Lalk, G., Shasha, D.: Efficient data reconciliation. Inf. Sci. 137(1-4), 1–15 (2001)
- Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: KDD, pp. 39–48 (2003)

References

- Christen,P.: Automatic record linkage using seeded nearest neighbour and support vector machine classification. In: KDD, pp. 151–159 (2008)
- Chen, Z., Kalashnikov, D.V., Mehrotra, S.: Exploiting context analysis for combining multiple entity resolution systems. In: SIGMOD, pp. 207–218 (2009)
- Ravikumar, P.D., Cohen, W.W.: A hierarchical graphical model for record linkage. In: UAI (2004)
- Bhattacharya, I., Getoor, L.: A latent dirichlet model for unsupervised entity resolution. In: SDM (2006)
- Sarawagi, S., Bhamidipaty, A.: Interactive deduplication using active learning. In: KDD (2002)
- Tejada,S., Knoblock,C.A., Minton,S.: Learning domain-independent string transformation weights for high accuracy object identification. In: KDD, pp. 350–359 (2002)
- Tejada, S., Knoblock, C.A., Minton, S.: Learning object identification rules for information integration. Inf. Syst.26(8), 607–633 (2001)
- Wang, J., Kraska, T., Franklin, M.J., Feng, J.: Crowder: Crowdsourcing entity resolution. PVLDB 5(11), 1483–1494 (2012)
- Verykios, V.S., Elmagarmid, A.K., Houstis, E.N.: Automating the approximate record-matching process. Inf.Sci. 126(1-4), 83–98 (2000)
- Weis, M., Naumann, F.: Detecting duplicates in complex xml data. In: ICDE, p. 109 (2006)
- Weis, M., Naumann, F.: Detecting duplicate objects in xml documents. In: IQIS, pp. 10–19 (2004)
- Leitao, L., Calado, P., Weis, M.: Structure-based inference of xml similarity for fuzzy duplicate detection. In:CIKM, pp. 293–302 (2007)
- Leitao,L., Calado,P., Herschel,M.: Efficient and effective duplicate detection in hierarchical data. IEEE Trans. Knowl. Data Eng. 25(5), 1028–1041 (2013)
- Herschel, M., Berti, L.: Application de mesures de distance pour la détection de problèmes de qualité de données. Book Chapter in La qualité et la gouvernance de données au service de la performance des entreprises, Ed. Hermes (2012)

References

- Puhlmann, S., Weis, M., Naumann, F.: Xml duplicate detection using sorted neighborhoods. In: EDBT, pp. 773–791 (2006)
- Bohm, C., de Melo, G., Naumann, F., Weikum, G.: Linda: distributed web-of-data-scale entity matching. In: CIKM, pp. 2104–2108 (2012)
- Papadakis, G., Ioannou, E., Niederee, C., Fankhauser, P.: Efficient entity resolution for large heterogeneous information spaces. In: WSDM, pp. 535–544 (2011)
- Papadakis, G., Ioannou, E., Palpanas, T., Niederee, C., Nejdl, W.: A Blocking Framework for Entity Resolution in Highly Heterogeneous Information Spaces. IEEE Trans. Knowl. Data Eng. (2013) (a). To appear
- Papadakis, G., Koutrika, G., Palpanas, T., Nejdl, W.: Meta-blocking: Taking entity resolution to the next level. IEEE Trans. Knowl. Data Eng. (2013) (b). To appear
- Hoffart, J., Seufert, S., Nguyen, D.B., Theobald, M., Weikum, G.: Kore: keyphrase overlap relatedness for entity disambiguation. In: CIKM, pp. 545–554 (2012)
- Ananthakrishna, R., Chaudhuri, S., Ganti, V.: Eliminating fuzzy duplicates in data warehouses. In: VLDB, pp. 586–597 (2002)
- Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Trans. Knowl. Data Eng. 19(1), 1–16 (2007)
- Getoor, L., Machanavajjhala, A.: Entity resolution: Theory, practice & open challenges. PVLDB 5(12), 2018– 2019 (2012)
- Papadakis, G., Demartini, G., Fankhauser, P., Karger, P.: The missing links: discovering hidden same-as links among a billion of triples. In: iiWAS, pp. 453–460 (2010)
- Hernández, M.A., Stolfo, S.J.: Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. Data Min. Knowl. Discov. 2(1): 9-37 (1998)

References

- Duan, S., Kementsietsidis, A., Srinivas, K., Udrea, O.: Apples and oranges: a comparison of rdf benchmarks and real rdf datasets. In: SIGMOD, pp. 145–156 (2011)
- Neumann, T., Moerkotte, G.: Characteristic sets: Accu-rate cardinality estimation for rdf queries with multiplejoins. In: ICDE, pp. 984–994 (2011)
- Rastogi, V., Dalvi, N.N., Garofalakis, M. N. : Large-Scale Collective Entity Matching. PVLDB 4(4): 208-218 (2011)
- Suchanek, F.M., Weikum, G.: Knowledge harvesting in the big-data era. In: SIGMOD, pp. 933-938 (2013)
- Hoffart, J., Suchanek F.M., Berberich, K., Weikum, G.: : YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia: Extended Abstract. In: IJCAI (2013)
- Whang,S.E., Marmaros, D., Garcia-Molina, H.: Pay-As-You-Go Entity Resolution. IEEE Trans. Knowl. Data Eng. 25(5): 1111-1124 (2013)
- Kolb, L., Thor, A., Rahm, E.:Don't match twice: redundancy-free similarity computation with MapReduce. In: Data Analytics in the Cloud (2013)
- [BM03] Bilenko, M., Mooney, R. J. , Cohen, W. W., Ravikumar, P. D., Fienberg, S. E.: Adaptive NameMatching in Information Integration. IEEE Intelligent Systems 18(5): 16-23 (2003)
- Baeza-Yates, R. A., Ribeiro-Neto, B. A.: Modern Information Retrieval. ACM Press / Addison-Wesley 1999, ISBN 0-201-39829-X
- Calado, P., Herschel, M., Leitão, L.: An Overview of XML Duplicate Detection Algorithms. Soft Computing in XML Data Management 2010: 193-224
- Christen, P.: Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Data-centric systems and applications, Springer 2012, ISBN 978-3-642-31163-5, pp. I-XIX, 1-270
- Jaro, M.A.. Advances in record linking methodology as applied to matching the 1985 census of Tampa Florida. J. Am. Stat. Assoc., vol. 84 (406), p. 414 - 420, 1989

References

- Kolb, L., Thor, A., Rahm, E.: Load Balancing for MapReduce-based Entity Resolution. In: ICDE (2012)
- Leser, U., Naumann, F.: Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen. dpunkt 2006, ISBN 3-89864-400-6
- McClellan, M.A.: Duplicate Medical Records: A Survey of Twin Cities Healthcare Organizations. AMIA Annual Symposium (2009)
- Naumann, F., Herschel, M.: An Introduction to Duplicate Detection. Synthesis Lectures on Data Management, Morgan & Claypool Publishers (2010)
- Weis, M., Naumann, F.: DogmatiX Tracks down Duplicates in XML. In: SIGMOD, pp. 431-442 (2005)
- Weis, M., Naumann, F., Jehle, U., Lufter, J., Schuster, H.: Industry-scale duplicate detection. PVLDB 1(2): 1253-1264 (2008)

Acknowledgements

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

- FP7 ICT IdeaGarden STREP http://idea-garden.org/
- GSRT ARISTEIA (LODGOV) Data Governance in the era of the Web of Data

License