Blocking Approaches

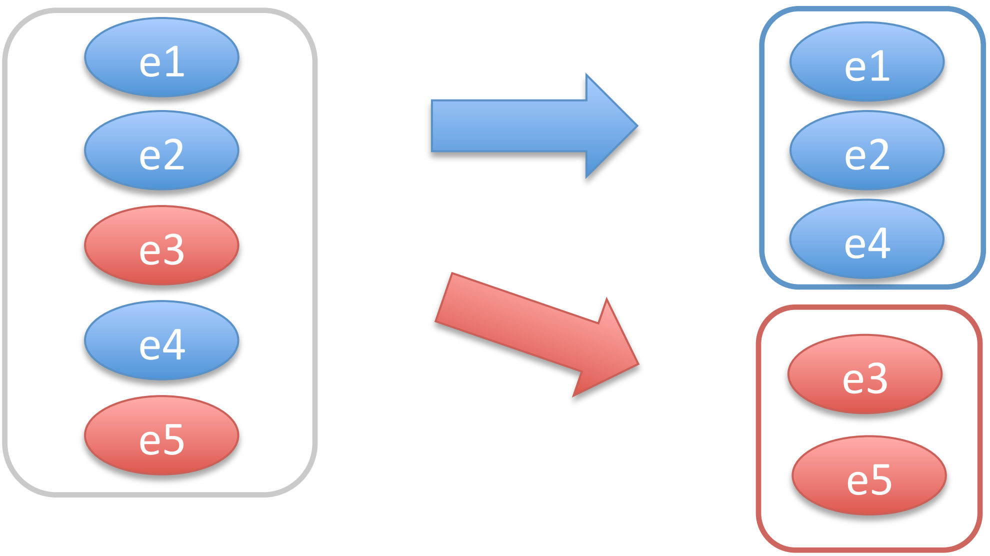
Blocking

To reduce the number of comparisons:

- Split entity descriptions into blocks
- Compare each description to the descriptions within the same block

Desiderata

- Similar entity descriptions in the same block
- Dissimilar entity descriptions in different blocks



Blocking Methodology

Blocking approaches rely on blocking keys

- Criteria on attributes, based on which the descriptions are placed into blocks

Given a blocking key:

The block in which a description will end up is determined by a similarity function on the value of the description for the blocking key

  - Blocking key value (BKV)

Using several blocking keys, places each description in many blocks

- Overlapping

Standard Blocking [Fellegi & Sunter 1969]

Entity descriptions with the same BKV end up in the same block

E.g. buildings located at the same place are put in the same block

	Name	Year	Architects	Location
e_1	Eiffel Tower	1889	Sauvestre	Paris
e_2	Statue of Liberty	1886	Bartholdi, Eiffel	NY
e_3	Lady Liberty		Eiffel	NY
e_4	Eiffel Tower	1889	Sauvestre	Paris
e_5	White Tower	1450		Thessaloniki

Standard Blocking [Fellegi & Sunter 1969]

Entity descriptions with the same BKV end up in the same block

E.g. buildings located at the same place are put in the same block

	Name	Year	Architects	Location
e_1	Eiffel Tower	1889	Sauvestre	Paris
e_2	Statue of Liberty	1886	Bartholdi, Eiffel	NY
e_3	Lady Liberty		Eiffel	NY
e_4	Eiffel Tower	1889	Sauvestre	Paris
e_5	White Tower	1450		Thessaloniki

Generated blocks (partition):

Paris	NY	Thessaloniki
e_1, e_4	e_2, e_3	e_5

Token Blocking [Papadakis et al. 2011]

Assume two clean sets E_1, E_2 of entity descriptions – *Clean-Clean Entity Resolution*

- Each distinct token t_i of each value of each description in $E_1 \cup E_2$ corresponds to a block
  - Each block contains all entities with the corresponding token
  - Pairs originating from the same (clean) set are not compared

Redundancy!

- The same pair of descriptions is contained in many blocks
- Many dissimilar pairs are put in the same block

Token Blocking - Example

E_1

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

E_2

about	Eiffel Tower
architect	Sauvestre
year	1889
located	Paris

e4

name	White Tower
location	Thessaloniki
year-constructed	1450

e5

Generated Blocks

Eiffel	Tower	Statue	Liberty	White	1889	Bartholdi
e_1, e_2, e_3, e_4	e_1, e_4, e_5	e_2	e_2, e_3	e_5	e_1, e_4	e_2

NY	Paris	1886	1450	Lady	Sauvestre	Thessaloniki
e_2, e_3	e_1, e_4	e_2	e_5	e_3	e_1, e_4	e_5

Token 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

Generated Blocks

Eiffel	Tower	~~Statue~~	Liberty	~~White~~	1889	~~Bartholdi~~
e_1, e_2, e_3, e_4	e_1, e_4, e_5	~~e_2~~	e_2, e_3	~~e_5~~	e_1, e_4	~~e_2~~

NY	Paris	~~1886~~	~~1450~~	~~Lady~~	Sauvestre	~~Thessaloniki~~
e_2, e_3	e_1, e_4	~~e_2~~	~~e_5~~	~~e_3~~	e_1, e_4	~~e_5~~

Blocks containing descriptions from only one collection are discarded

Token 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

Generated Blocks

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

NY	Paris	Sauvestre
e_2, e_3	e_1, e_4	e_1, e_4

The pair (e_1, e_4) is contained in 5 different blocks!

Token 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

Generated Blocks

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

NY	Paris	Sauvestre
e_2, e_3	e_1, e_4	e_1, e_4

Redundant comparisons are performed between (e_1, e_3), (e_2, e_4), (e_1, e_5)

Token blocking achieves:

High recall at the cost of low precision and low efficiency:

- Most true matches are placed in the same block
- Many non-matches are also placed in the same block
- The same pair of descriptions is contained in many blocks

Token blocking totally ignores the valuable information of attribute names

Can we exploit the way data are published on the Web?

Many URIs contain semantics

- Use them as indications of matches between descriptions

[Papadakis et al. 2010]

E.g. 66% of the 182 million URIs of BTC09 follow the scheme: Prefix-Infix(-Suffix)

- Prefix describes the source, i.e. domain, of the URI
- Infix is a local identifier
- The optional Suffix contains details about the format, e.g. .rdf and .nt, or a named anchor

http://km.aifb.kit.edu/pro jects/btc-2009/

Prefix-Infix(-Suffix) [Papadakis et al. 2012]

Token blocking on the Infixes/literals appearing in the values of descriptions

http://en.wikipedia.org/wiki/Linked_data#Principles

- Prefix: describes the source (domain)
- Infix: local identifier
- Suffix (optional): details about the format, or a named anchor

Techniques:

Infix blocking

- The blocking key is the infix of the URI of the entity description

Infix profile blocking

- The blocking keys are the infixes in the values of each entity description

Infix Blocking

The blocking key is the infix of the URI of the entity description

yago:Statue_of_Liberty

skos:prefLabel	Statue of Liberty
yago:isLocatedIn	yago:Liberty_Island **e1**

dbpedia:Statue_of_Liberty

rdfs:label	Statue of Liberty
dbprop:location	dbpedia:Liberty_Island **e2**

fb:m.072p8

fb:official_name	Statue of Liberty
fb:contained_by	fb:m.026kp2
ex:location	ex:Liberty_Island **e3**

geonames:5139572

geonames:name	Statue of Liberty
geonames:nearby	geonames:5124330 **e4**

yago:Tina_Brown

skos:prefLabel	Tina Brown
yago:linksTo	yago:Liberty_Island **e5**

Generated blocks:

Statue_of_Liberty
e_1, e_2

m.072p8
e_3

5139572
e_4

Tina_Brown
e_5

Infix Profile Blocking

The blocking keys are the infixes in the values of each entity description

skos:prefLabel	Statue of Liberty
yago:isLocatedIn	yago:Liberty_Island

e1

rdfs:label	Statue of Liberty
dbprop:location	dbpedia:Liberty_Island

e2

fb:official_name	Statue of Liberty
fb:contained_by	fb:m.026kp2
ex:location	ex:Liberty_Island

e3

geonames:name	Statue of Liberty
geonames:nearby	geonames:5124330

e4

skos:prefLabel	Tina Brown
yago:linksTo	yago:Liberty_Island

e5

pros: (e1, e3) correctly identified
cons: (e1, e5) mistakenly identified

Generated blocks:

Liberty_Island	m.026kp2	5124330
e_1, e_2, e_3, e_5	e_3	e_4

Drawback!
The effectiveness of these approaches relies on the good naming practices of the data

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	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

Block size threshold = 3

Removing Oversized Blocks

Done!

~~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

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

HOW?

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]

Match_likelihood(e_i, e_j) = Jaccard(blocks(e_i), blocks(e_j)) =
|blocks(e_i) ∩ blocks(e_j)| / |blocks (e_i) ∪ 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	STEP 2
Block Building	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

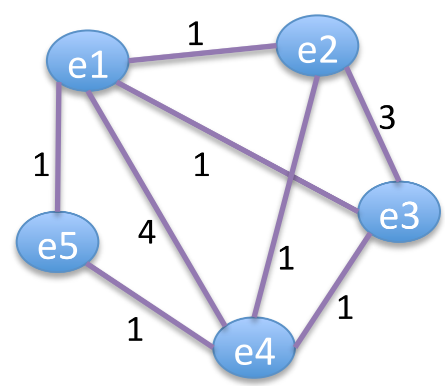
Blocks:
(with token blocking)

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

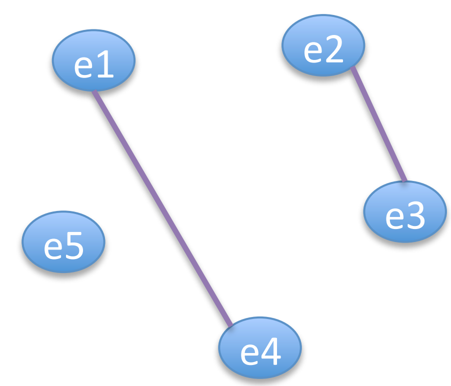
13 comparisons
to identify 2 matches

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	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}

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
e_1, e_4, e_{14}

1886
e_2, e_5, e_{25}

1885
e_3

P
e_{14}

N
e_{25}

L
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	1886	1885
e_1, e_4, 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}

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

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!*

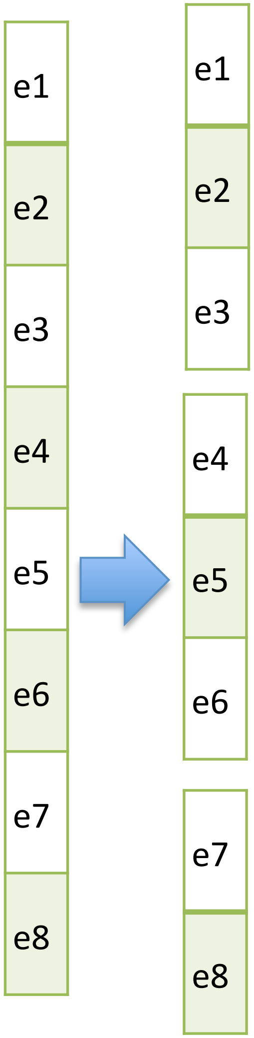
The map phase reflects blocking (re-distribute descriptions)

The reduce phase reflects entity resolution (check for matches)

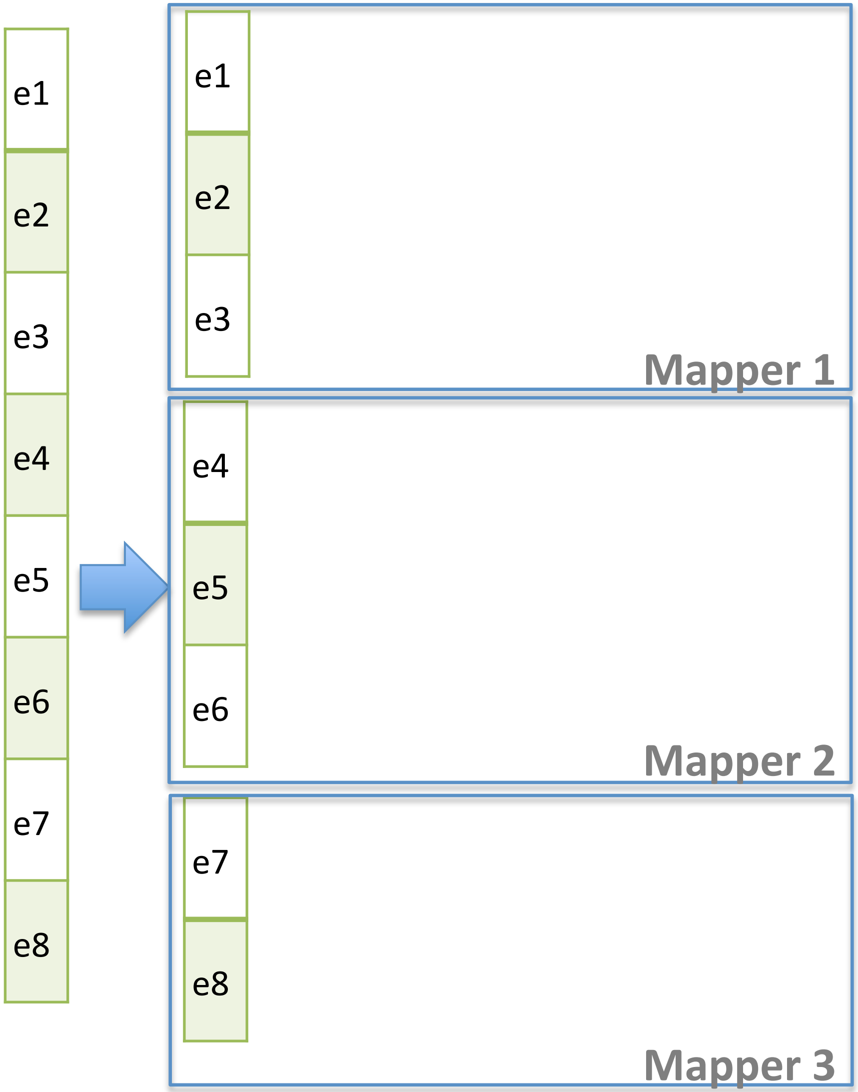
MapReduce – Input Data

e1
e2
e3
e4
e5
e6
e7
e8

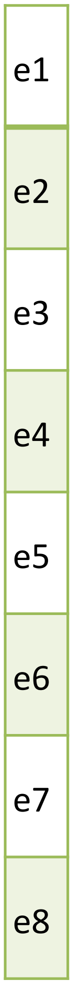
MapReduce – Input Data Partitioning

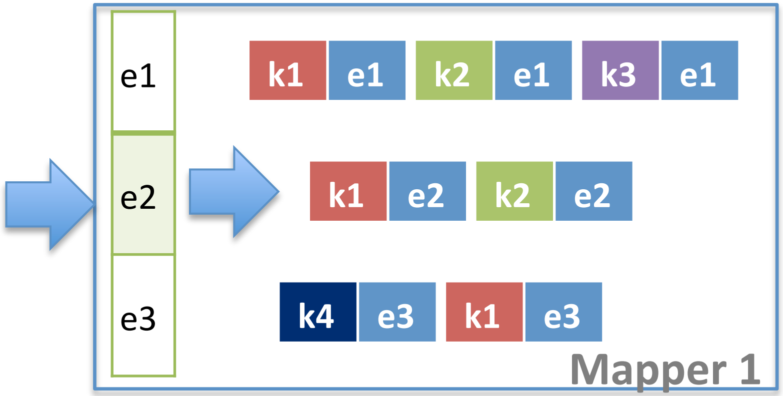


MapReduce – Mapper Input



MapReduce – Mapper Example





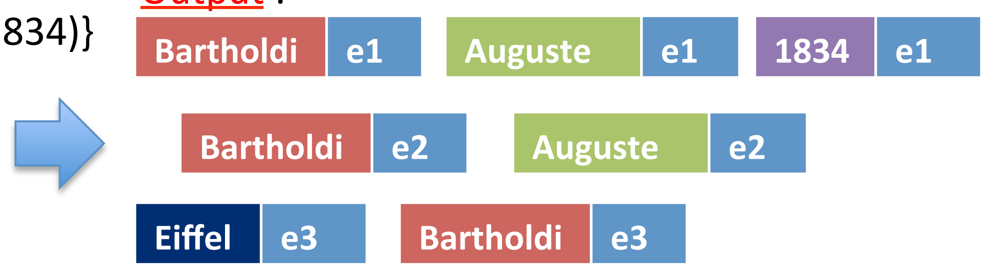
Input :

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

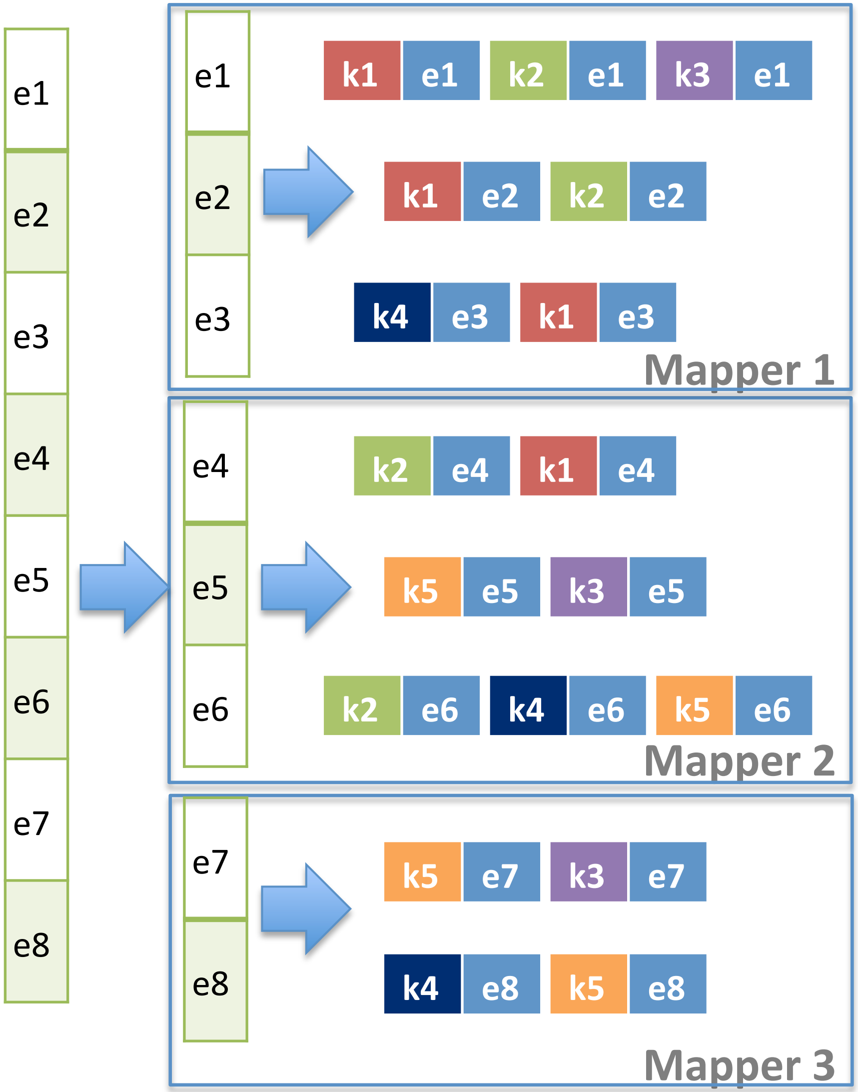
e2={(about, Auguste Bartholdi)}

e3={(architects, Bartholdi Eiffel)

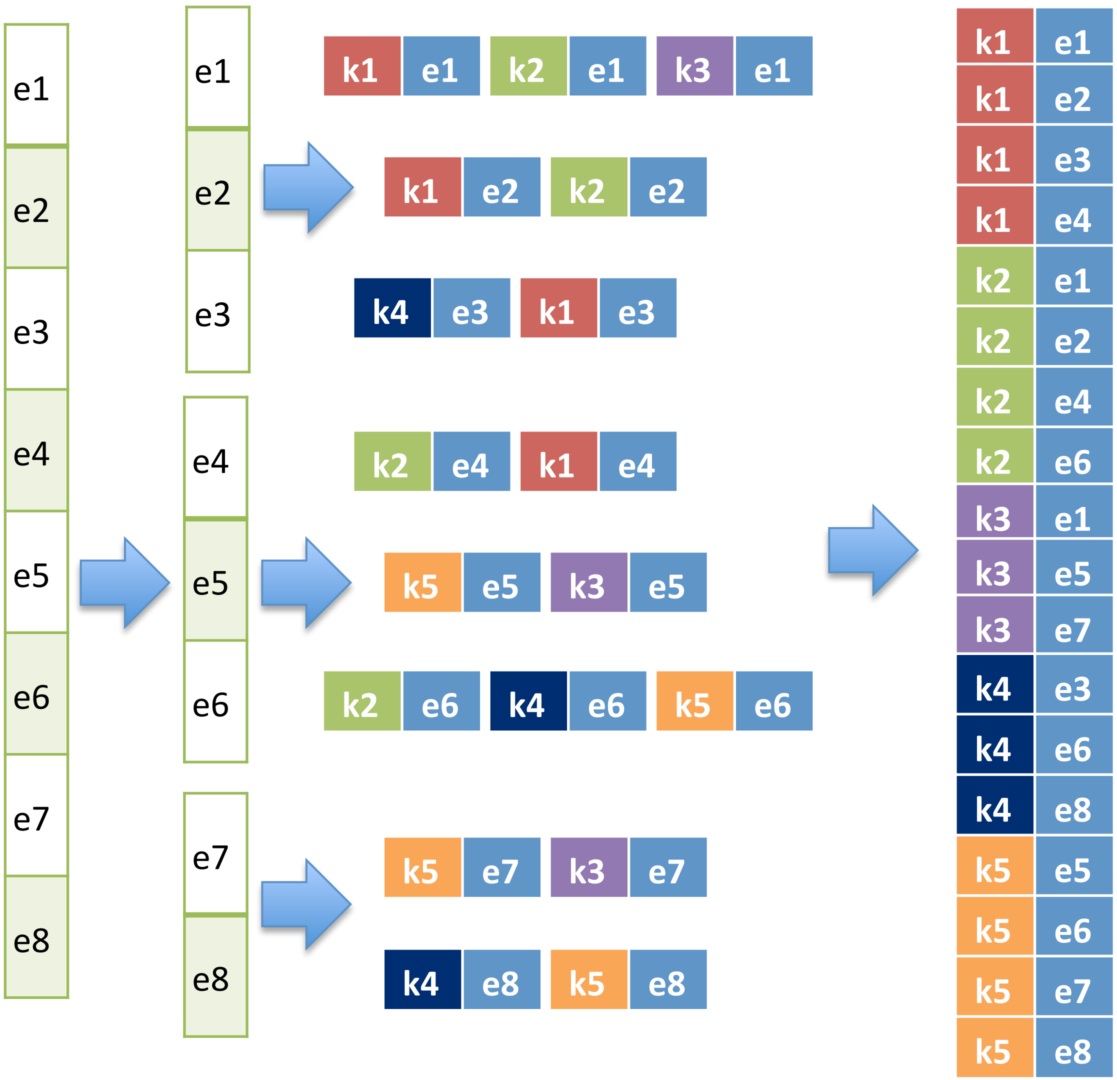
Output :



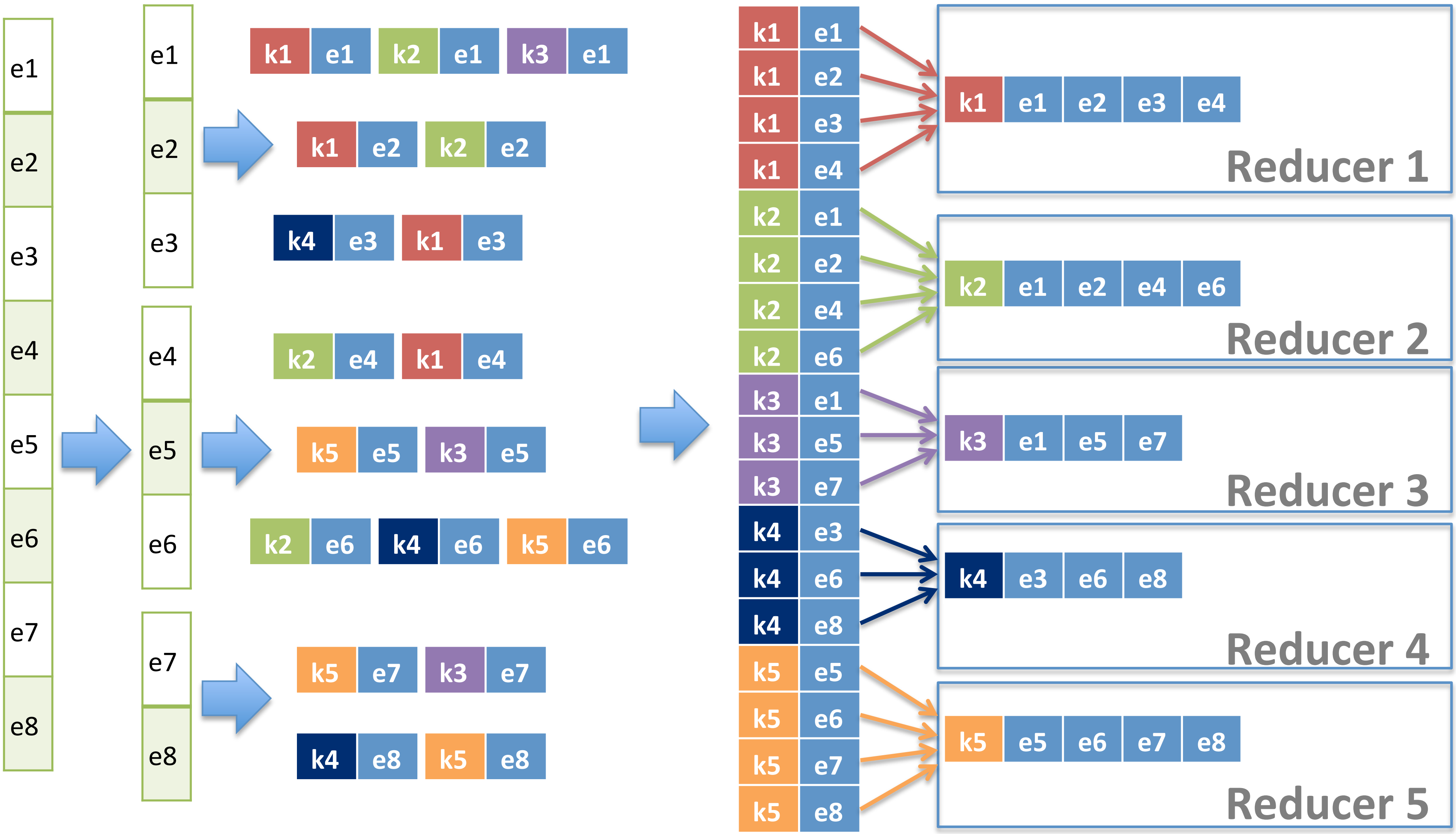
MapReduce – Mapper Output



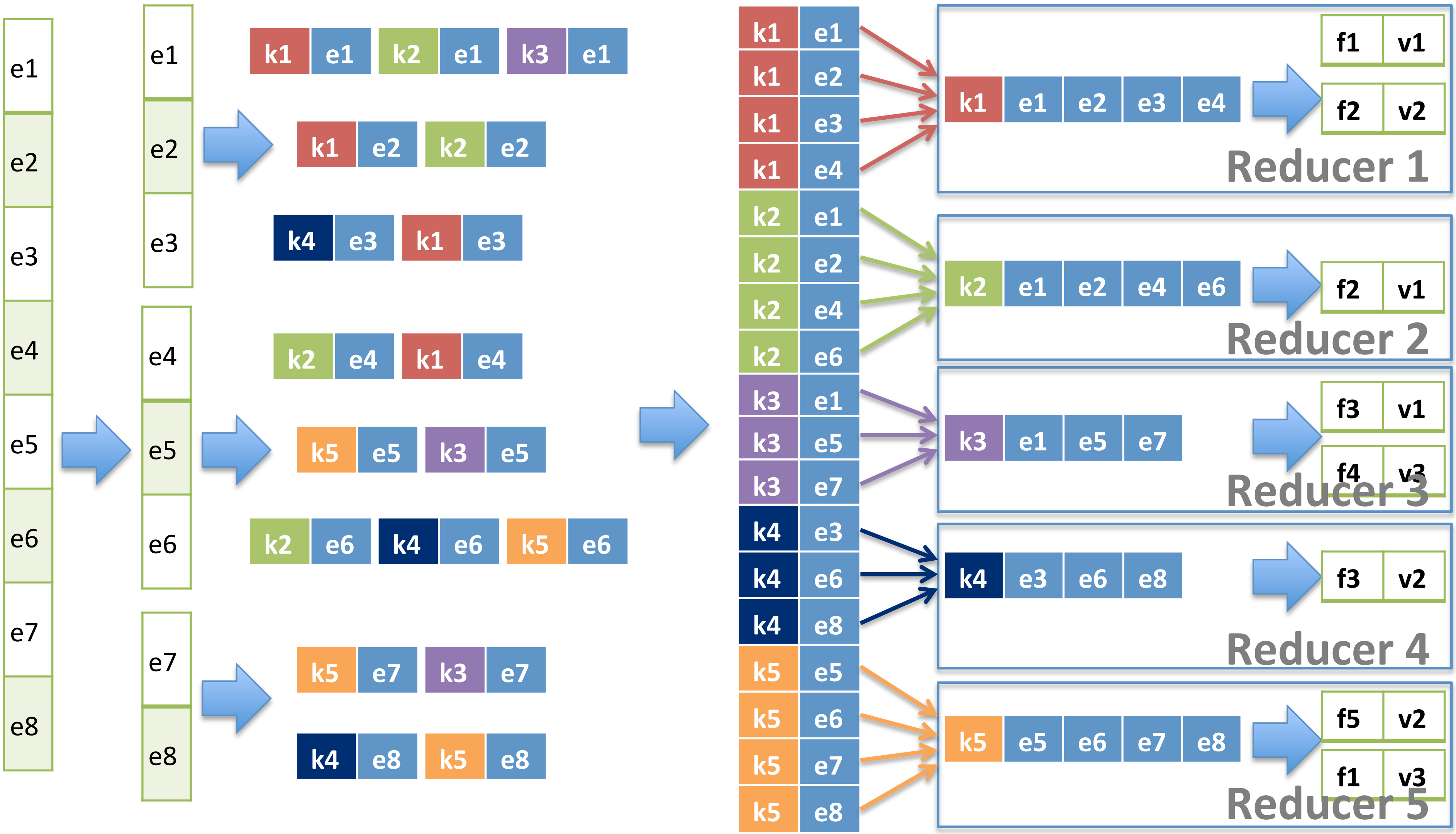
MapReduce – Shuffling & Sorting



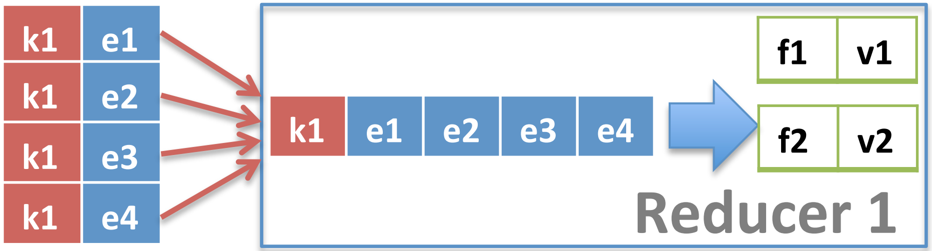
MapReduce – Merging

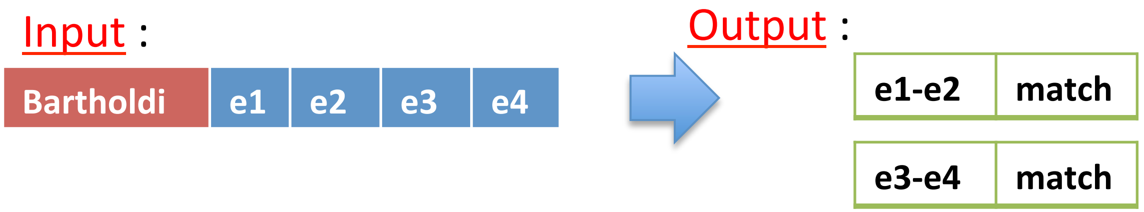


MapReduce – Reducer



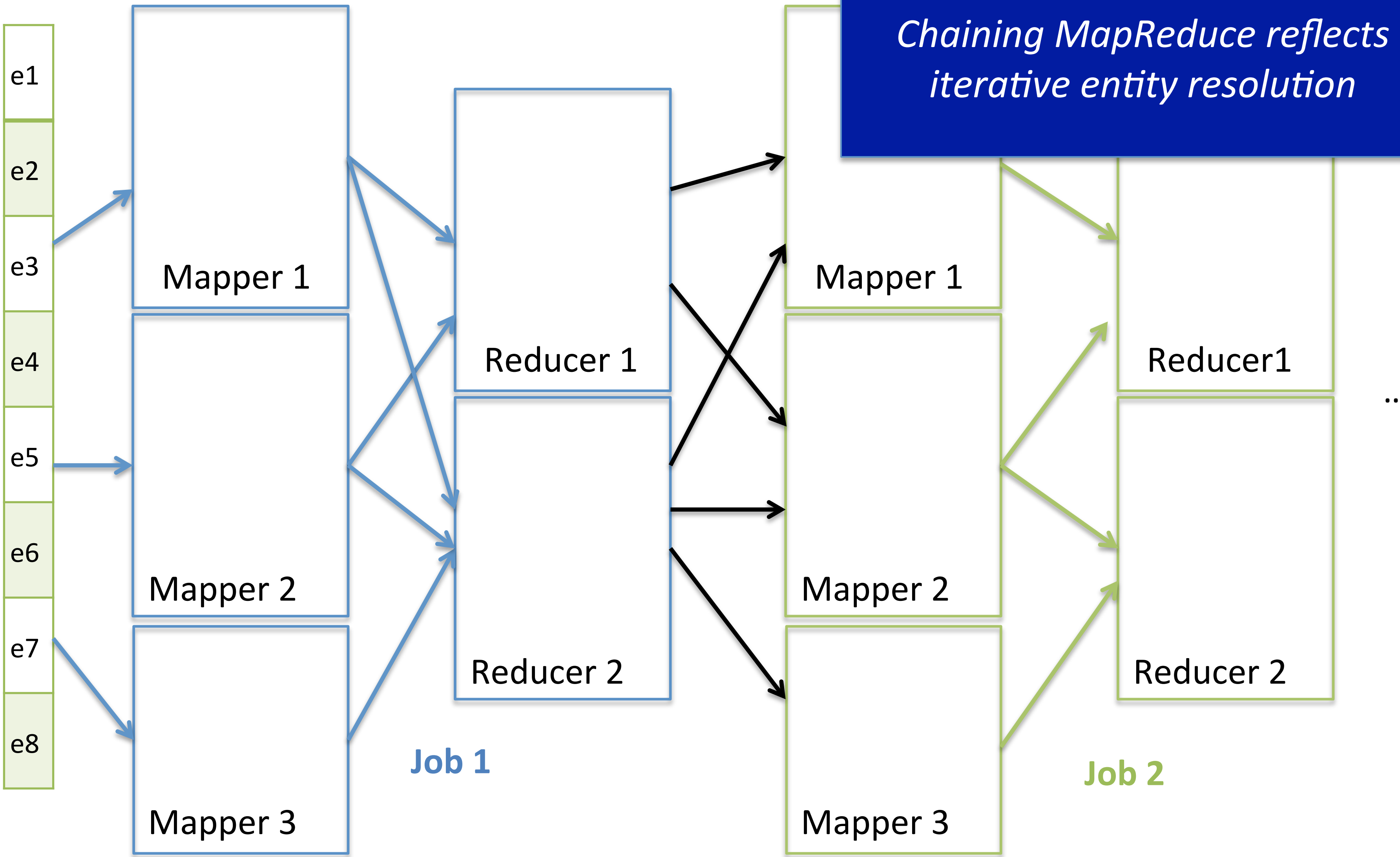
MapReduce – Reducer Example





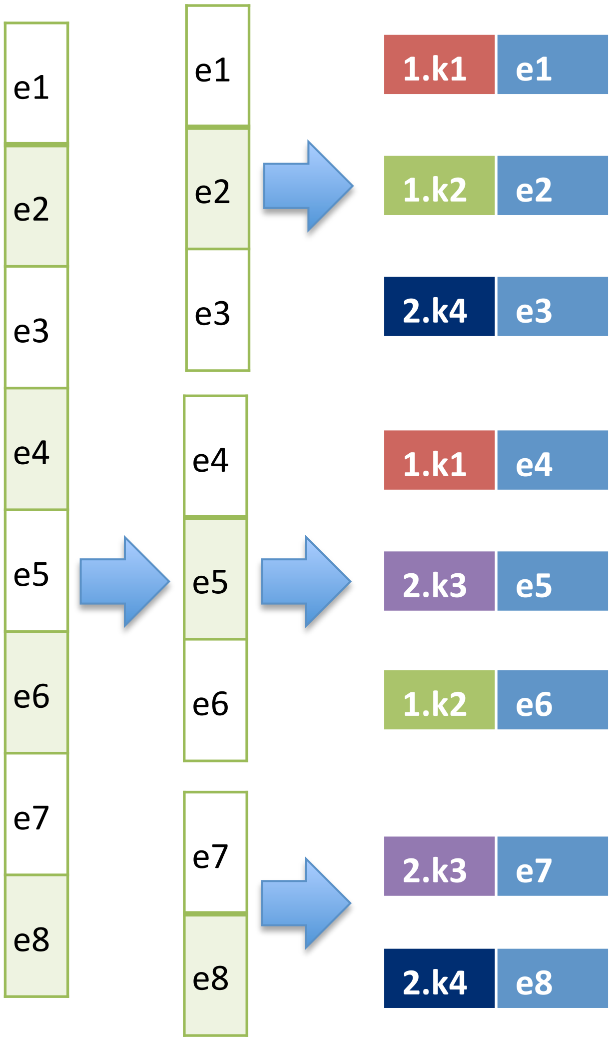
Chaining MapReduce Jobs

Chaining MapReduce reflects iterative entity resolution



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

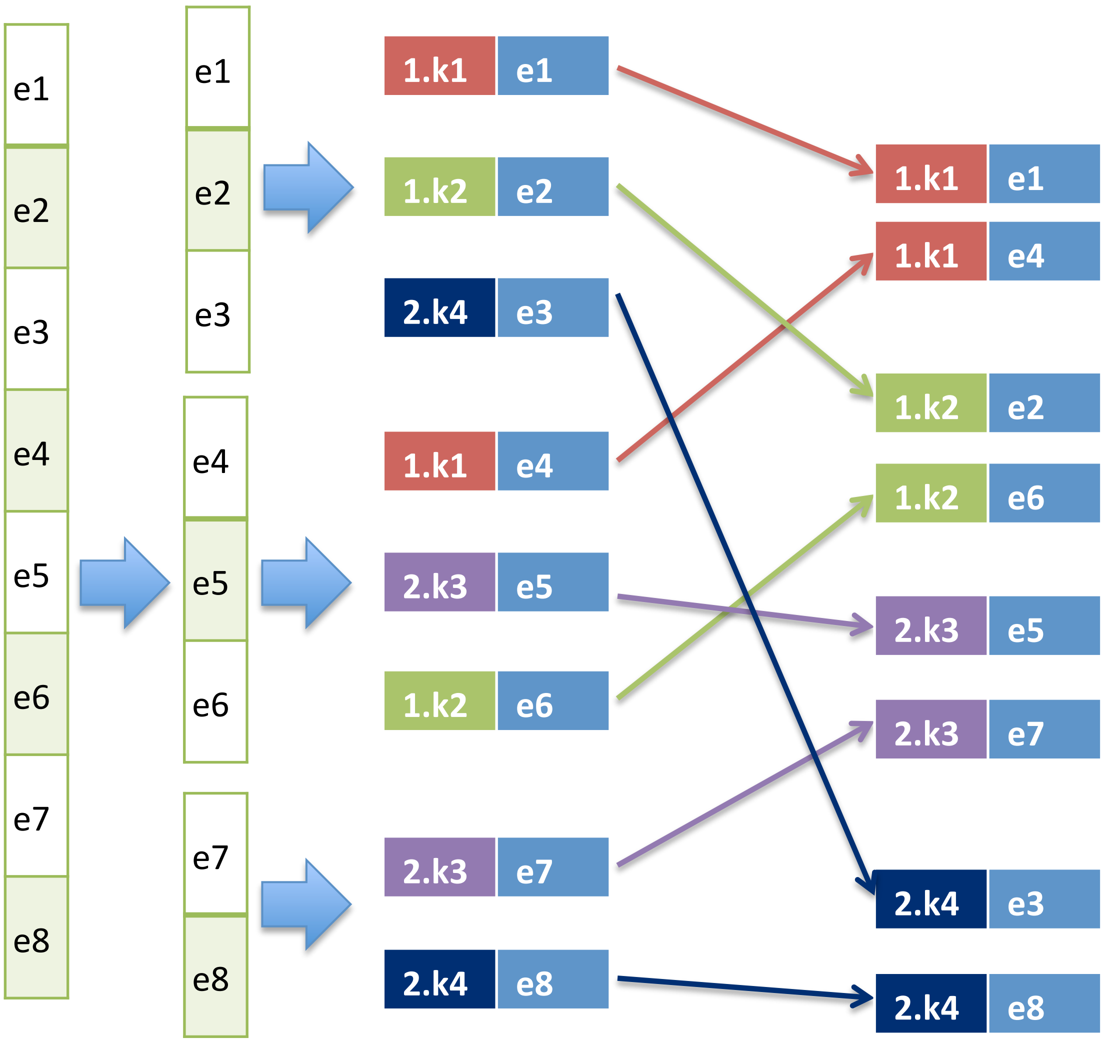
Dedoop – Sorted Neighborhood [Kolb et al. 2011]



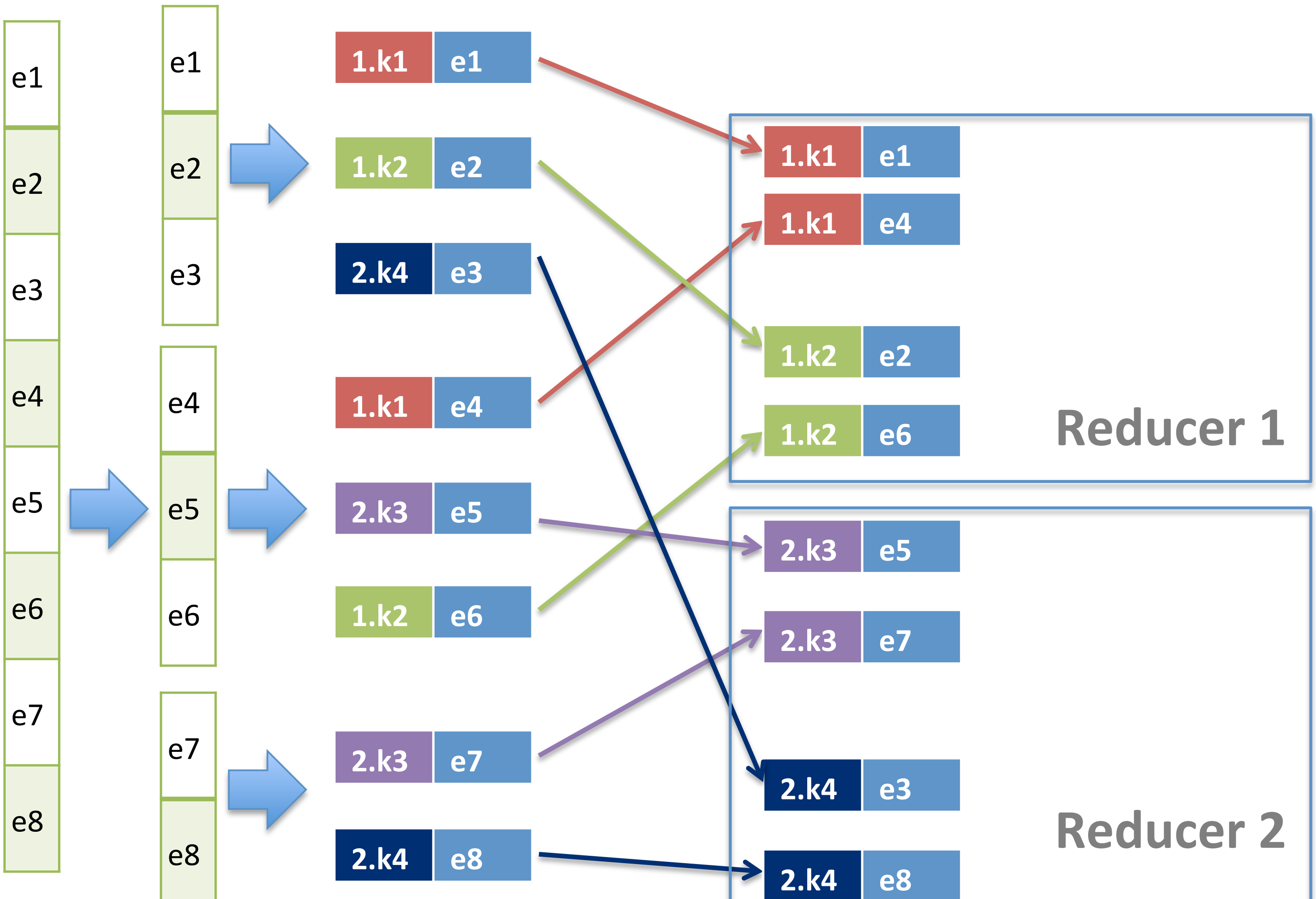
composite key = (partitionID, BKV)
partitionID(BKV) = 1, if BKV < "k3"
partitionID(BKV) = 2, else

(we know that we have two reducers available)

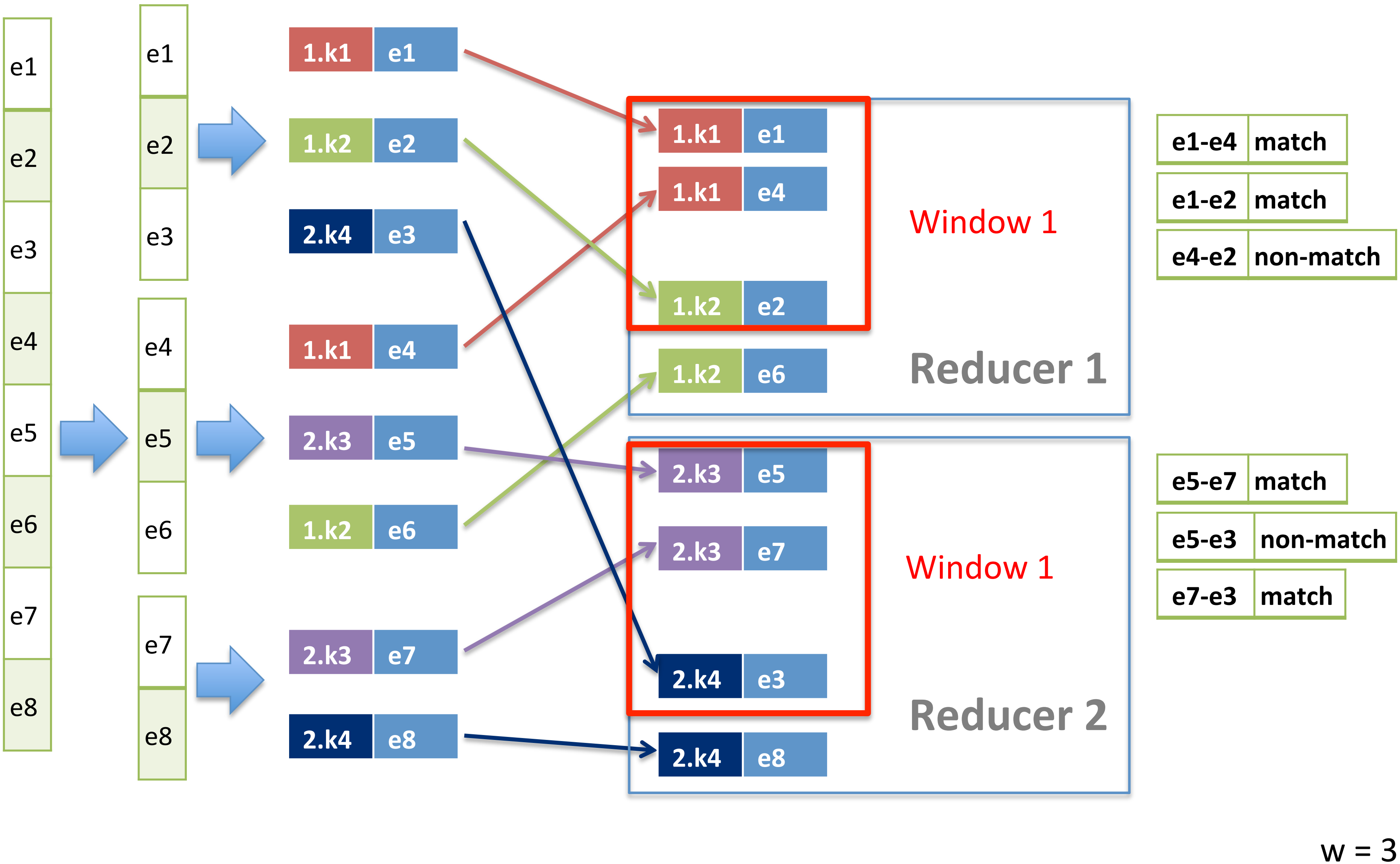
Dedoop SN: Sorting the Keys



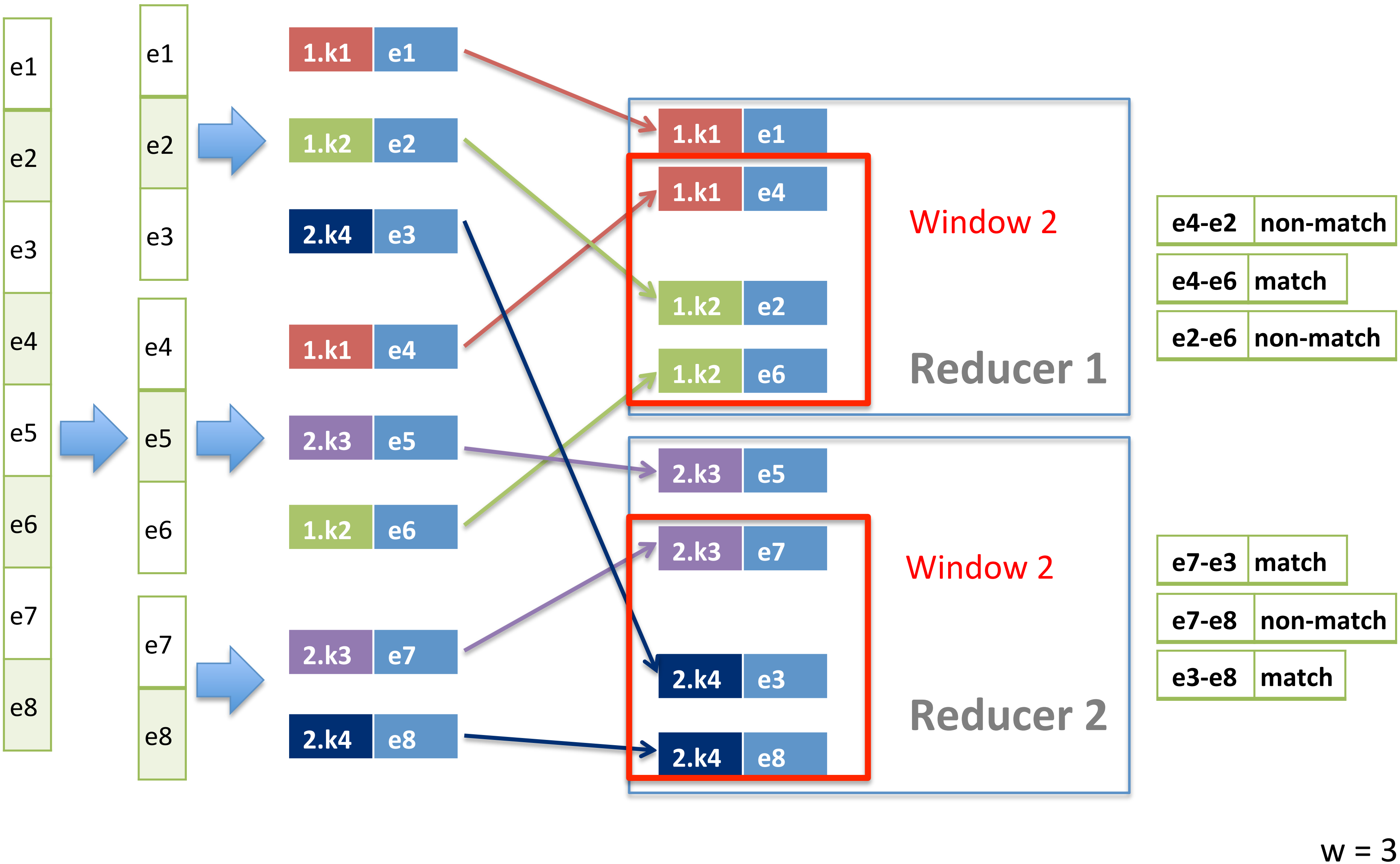
Dedoop SN: Reducers Apply the Sliding Window



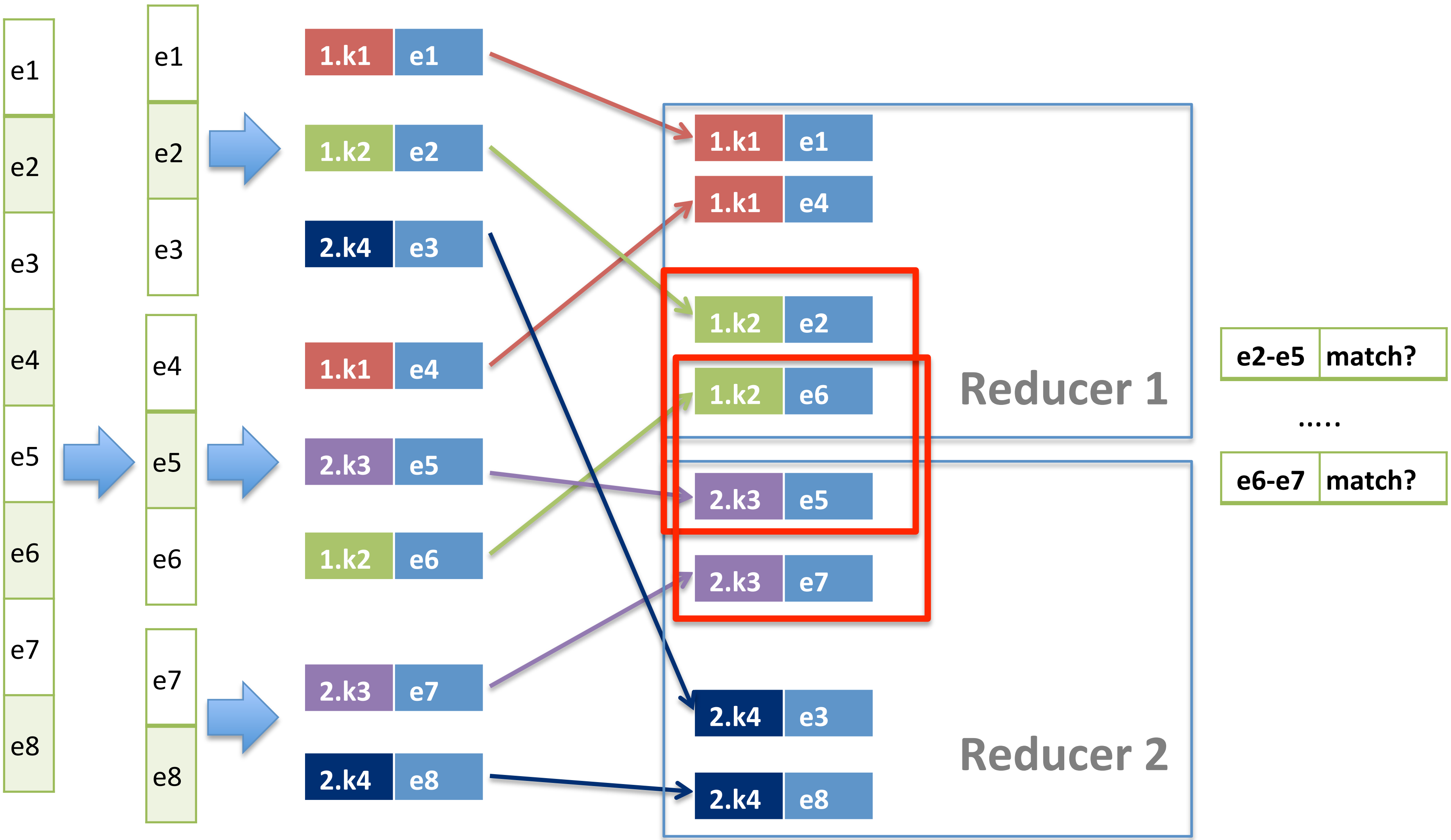
Dedoop SN: Reducers Apply the Sliding Window



Dedoop SN: Reducers Apply the Sliding Window



Dedoop SN: We Also Need To Compare The Boundary Entities



w = 3

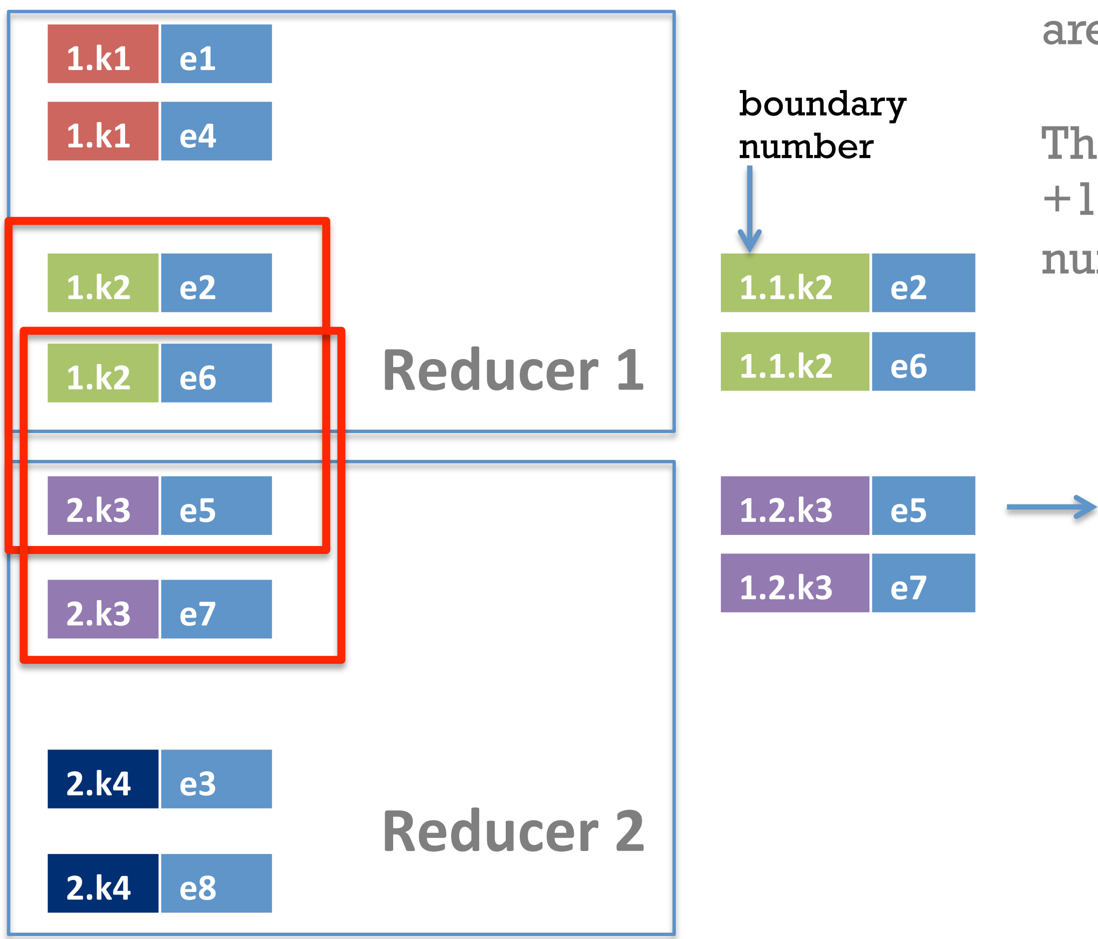
Dedoop SN: Reducers Also Output the Boundary Descriptions

Add a <u>boundary number prefix</u> to the output composite keys

<u>Boundary number:</u>

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

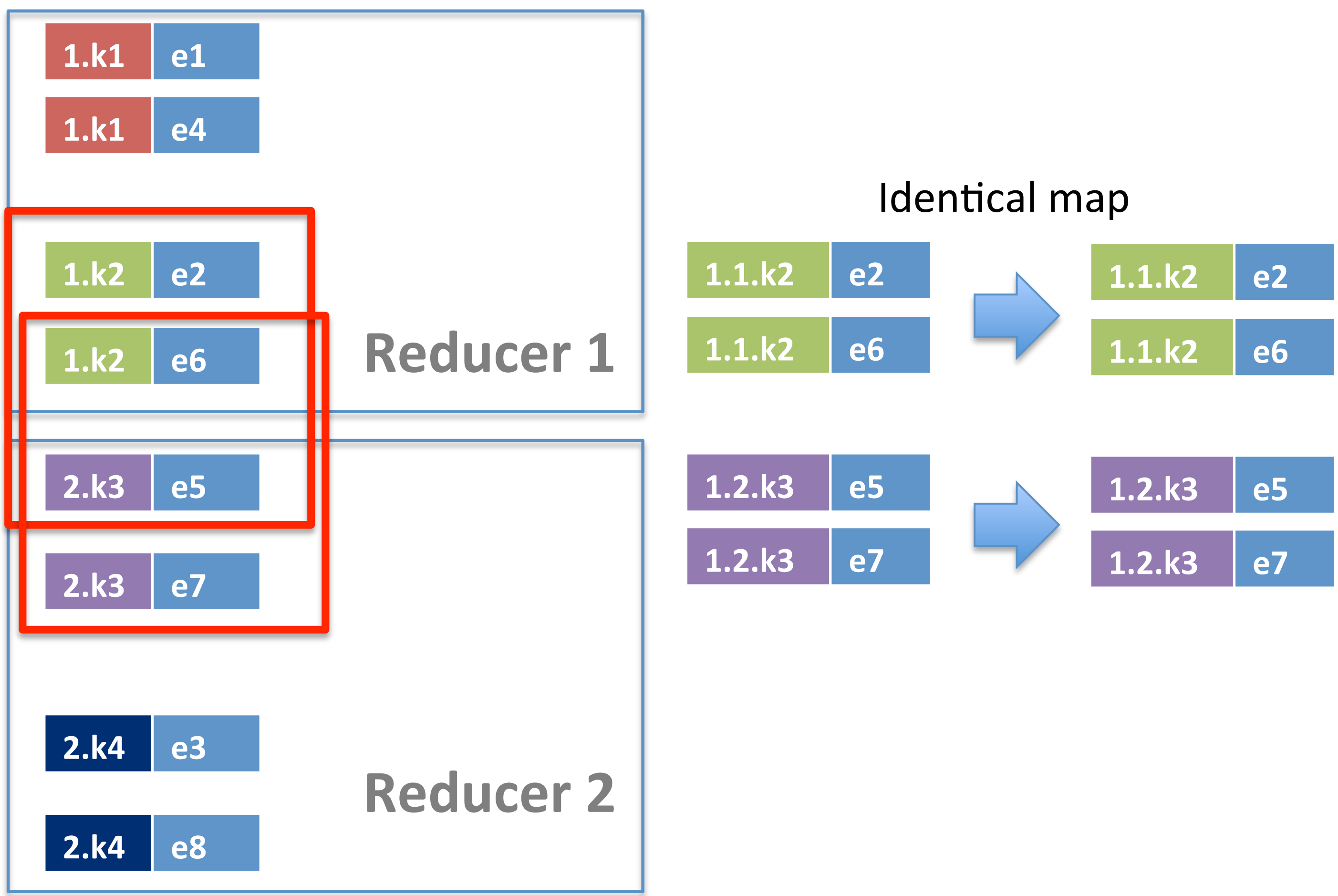
The first w-1 descriptions of reducer i +1 are also assigned the boundary number i



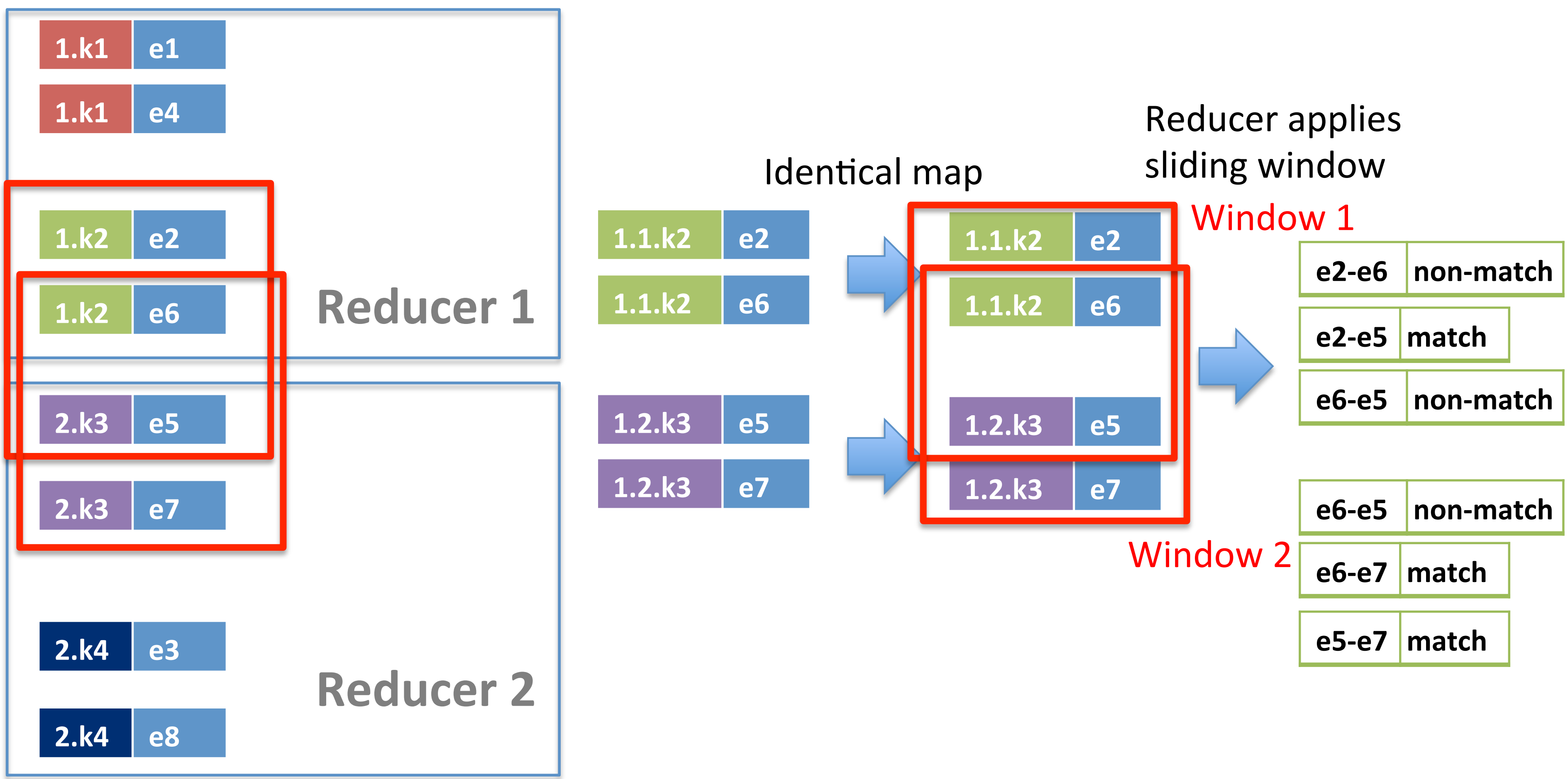
The actual blocking key of e5 is k3, it was assigned to reducer 2 and it is associated with boundary number 1

w = 3

Dedoop SN: New MapReduce Job for the Boundary Pairs

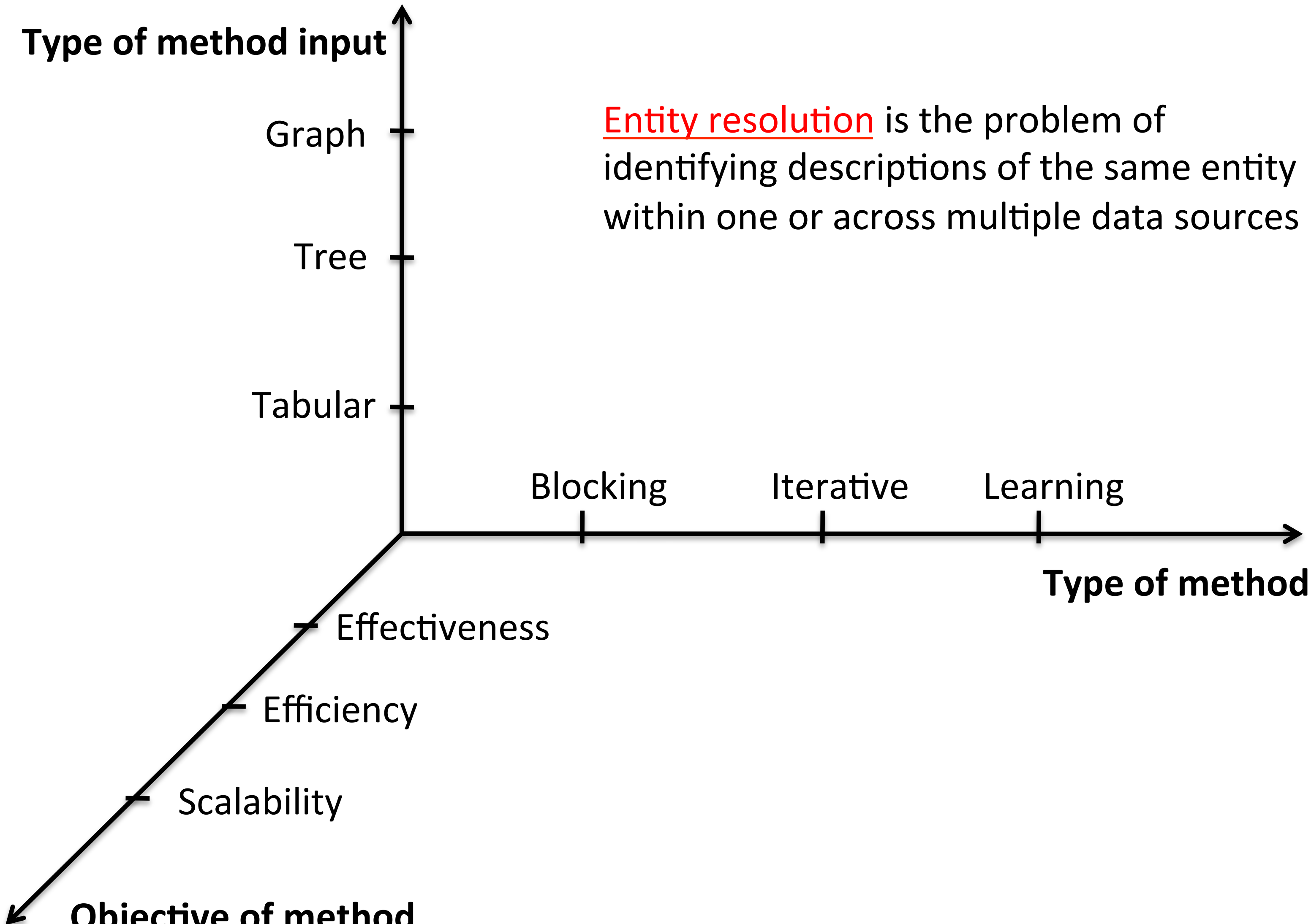


Dedoop SN: Partition by Boundary Number



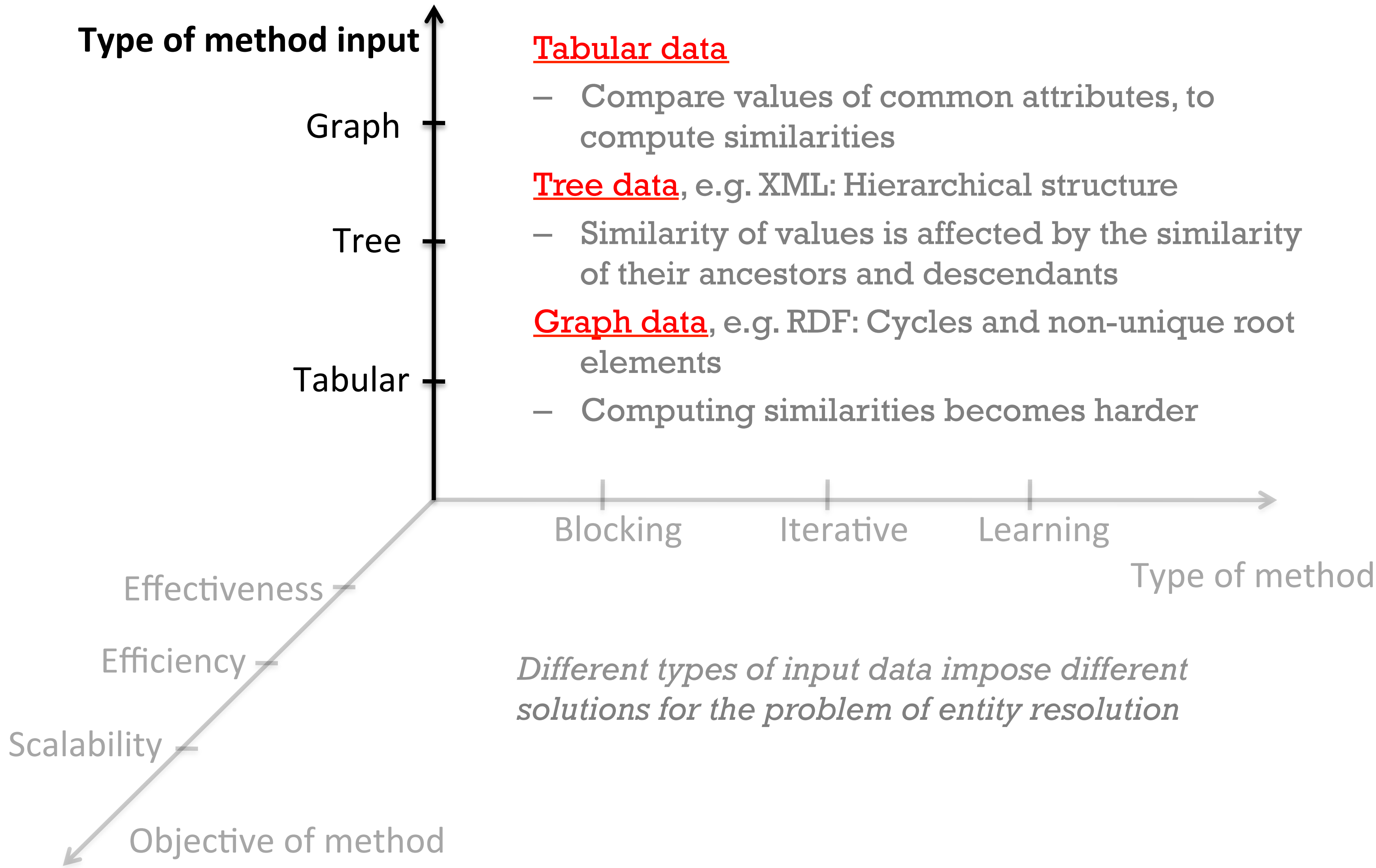
Conclusions

Solution Space



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

Solution Space



<u>Tabular data</u>

- Compare values of common attributes, to compute similarities

<u>Tree data</u>, e.g. XML: Hierarchical structure

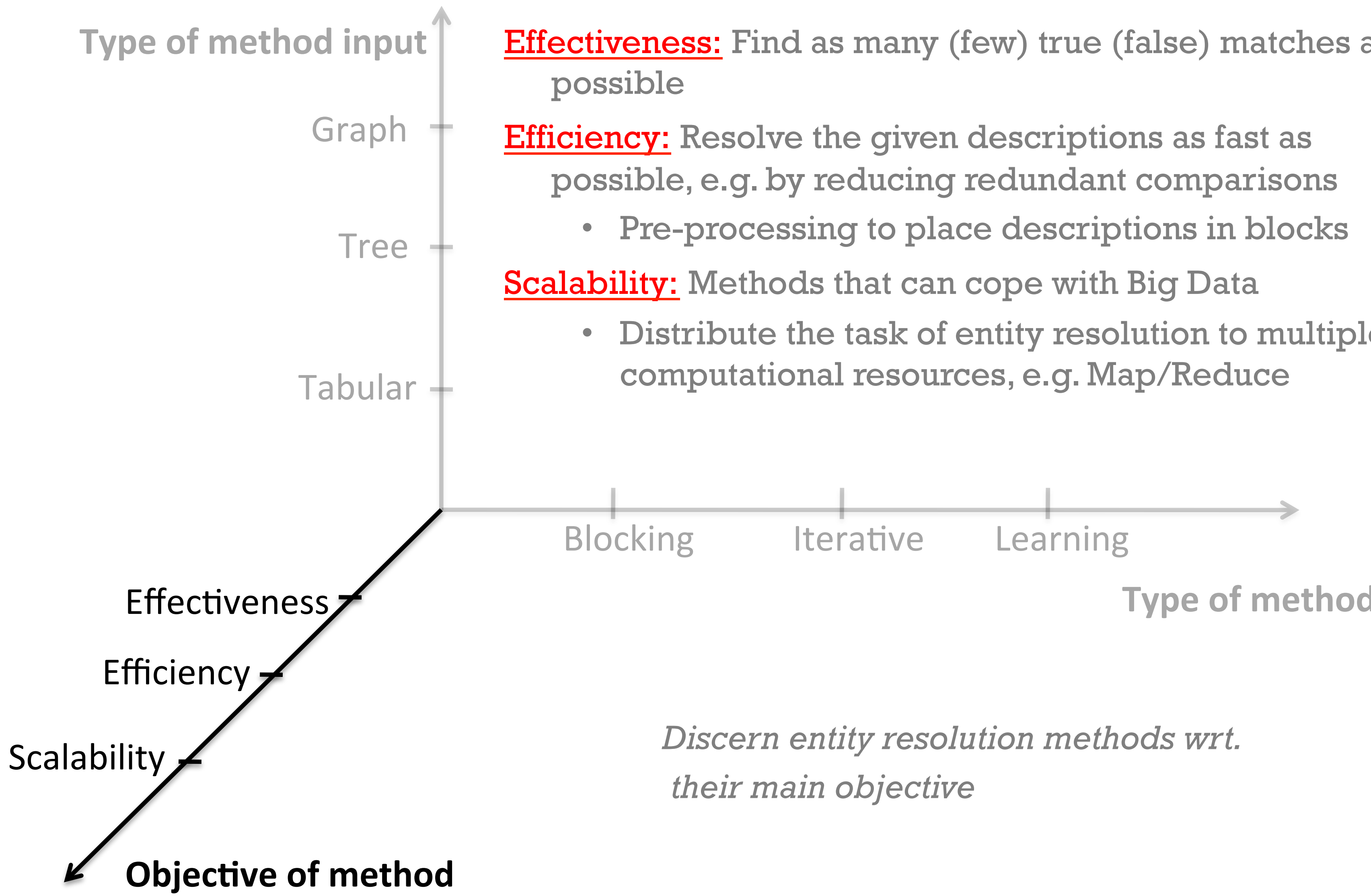
- Similarity of values is affected by the similarity of their ancestors and descendants

<u>Graph data</u>, 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

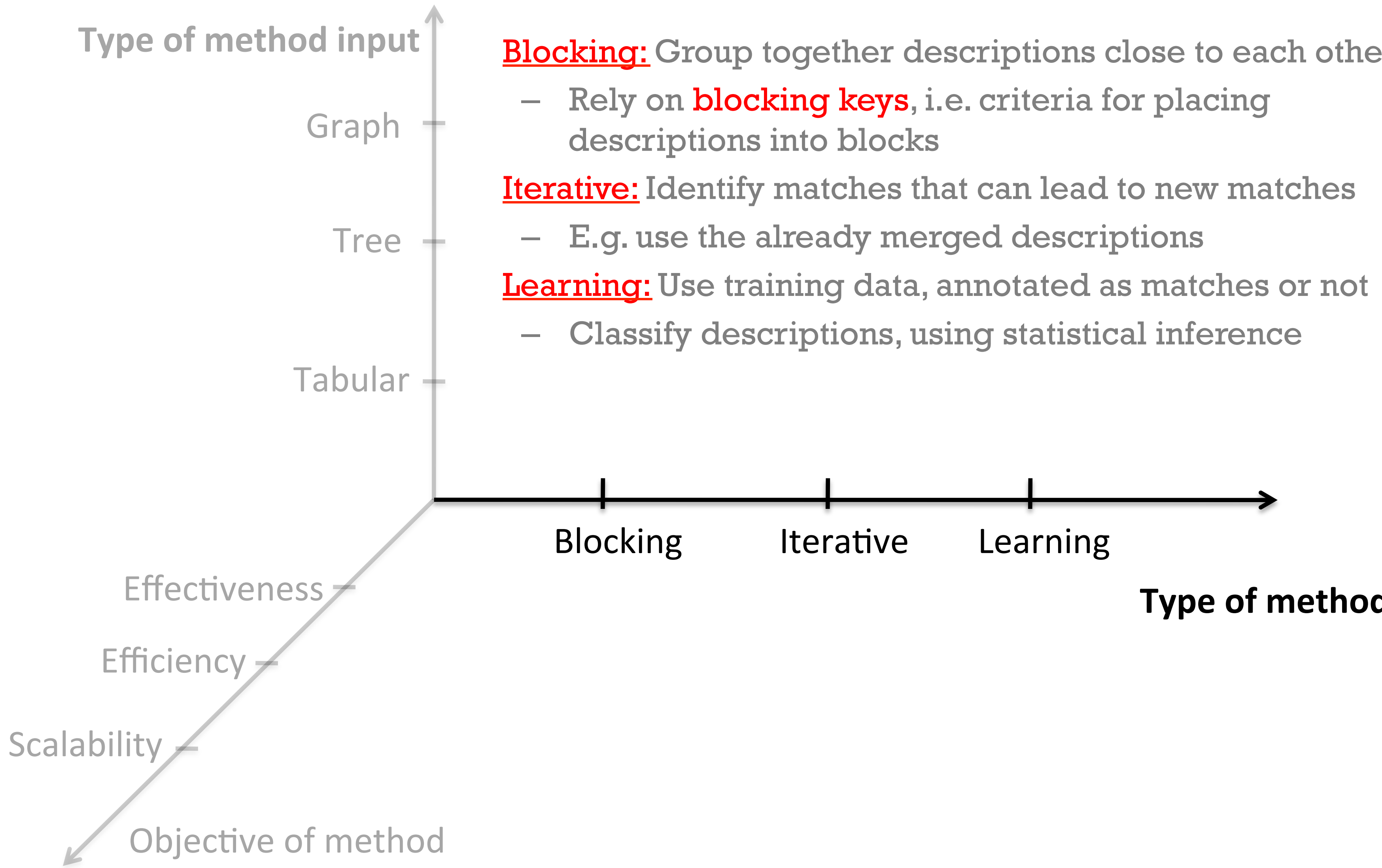
– Rely on **blocking keys**, i.e. criteria for placing descriptions into blocks

Iterative: Identify matches that can lead to new matches

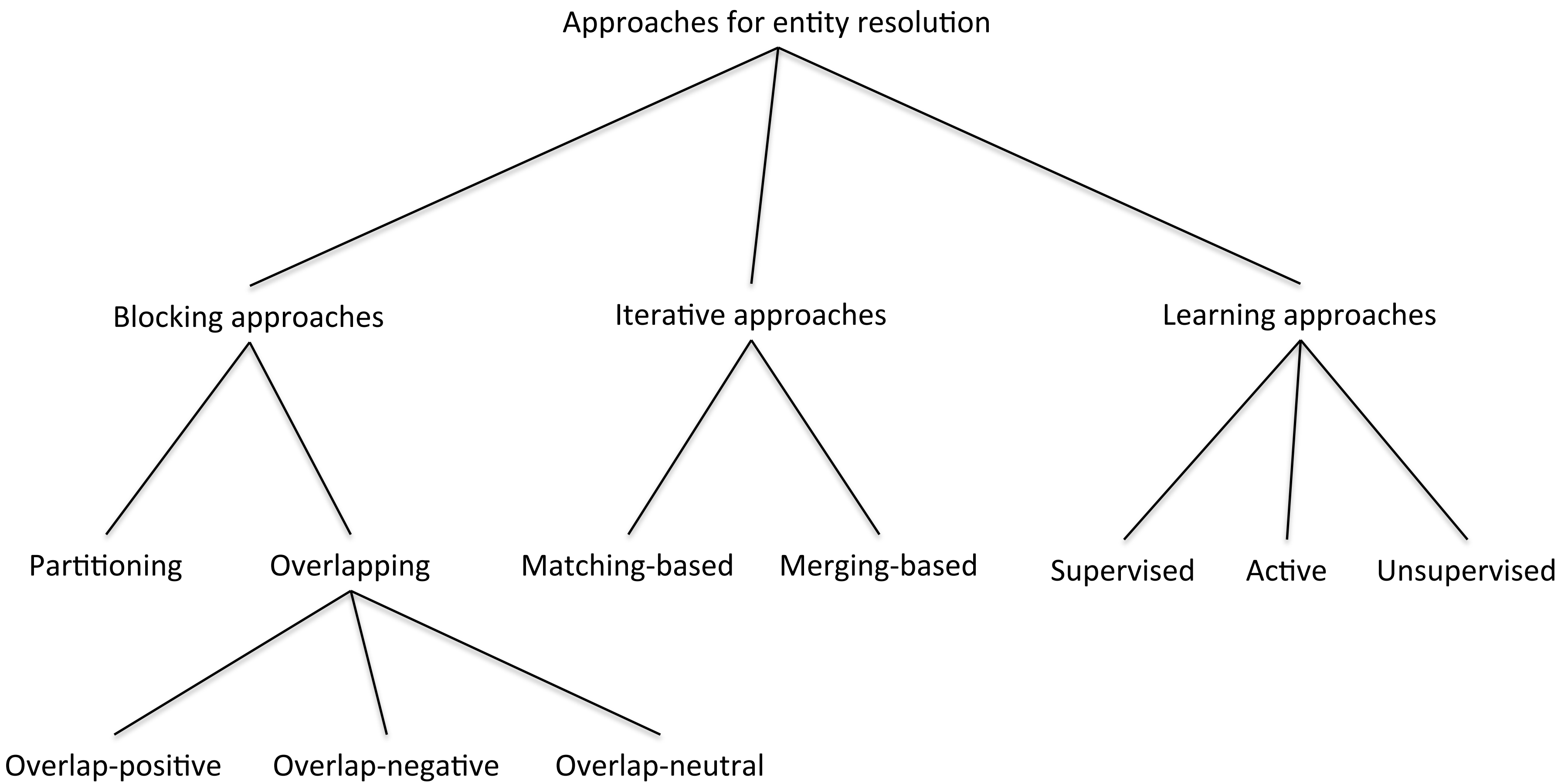
– E.g. use the already merged descriptions

Learning: Use training data, annotated as matches or not

– Classify descriptions, using statistical inference



Solution Space – A Detailed Taxonomy



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		• +	