International World Wide Web Conference
Seoul, Korea
April 7 - 11, 2014
WWW 2014

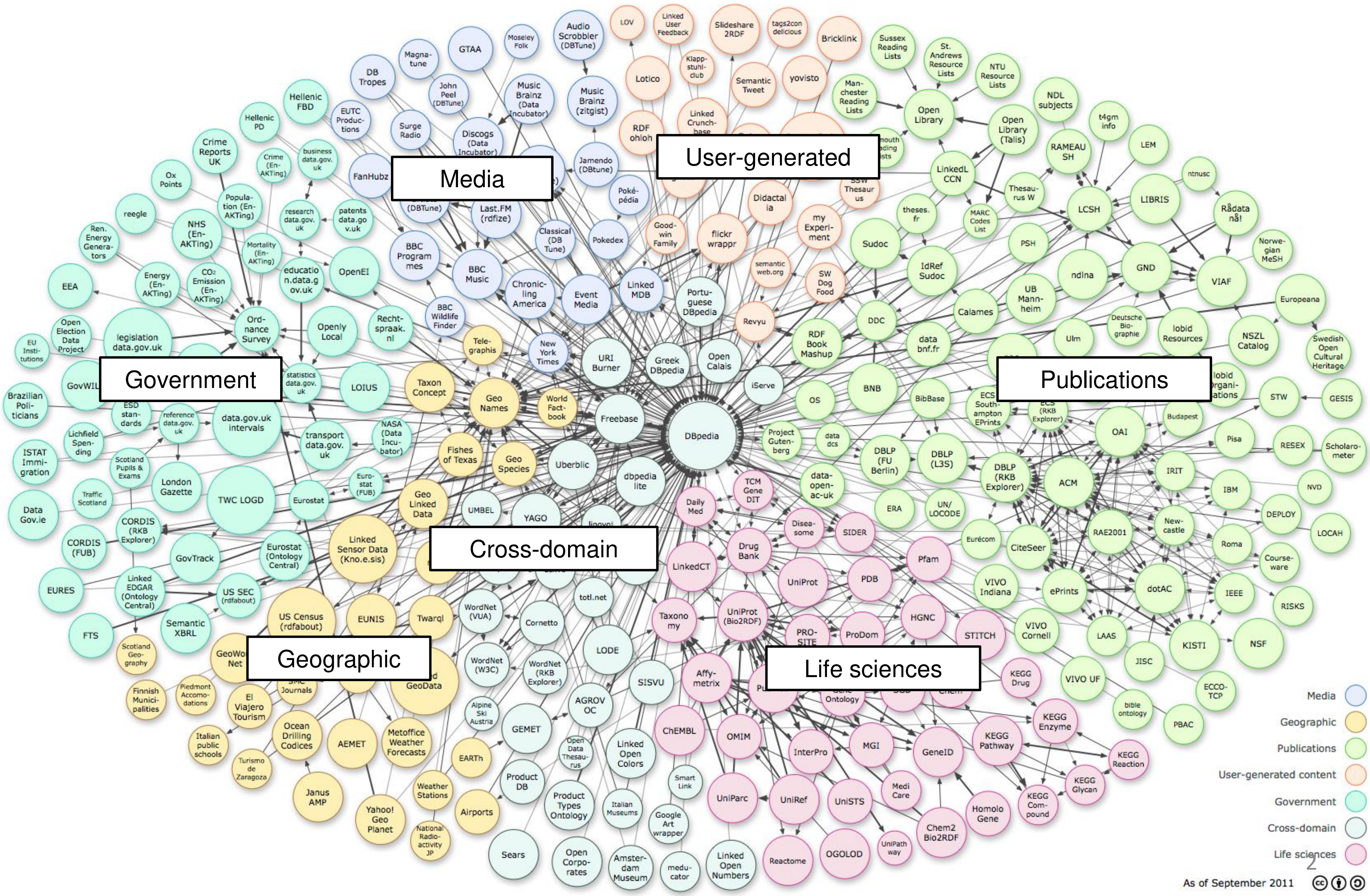
Entity Resolution in the Web of Data

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

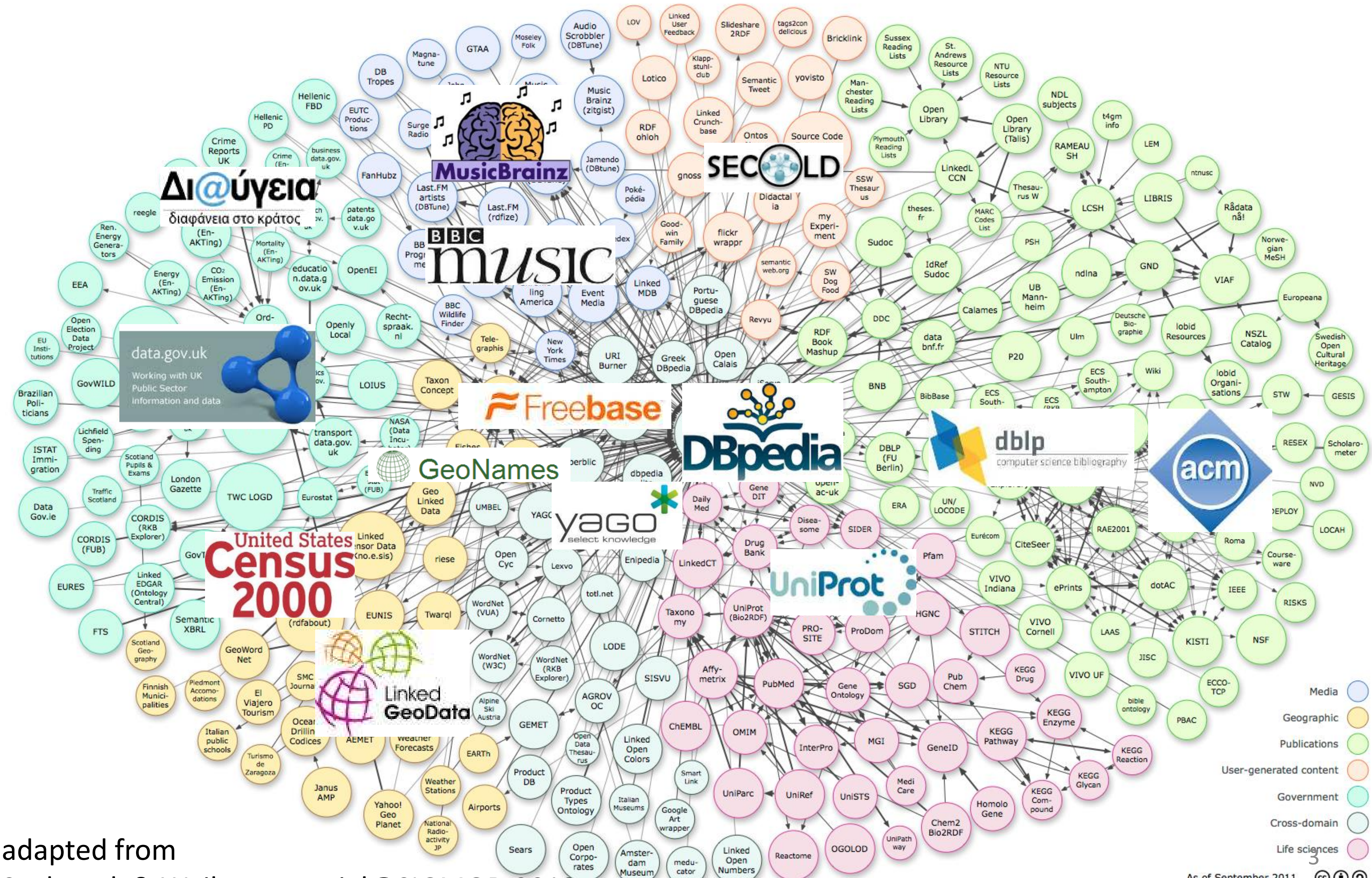
kstef@ics.forth.gr, vefthym@ics.forth.gr, melanie.herschel@lri.fr
vassilis.christophides@technicolor.com

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

LOD Cloud and the Web of Data

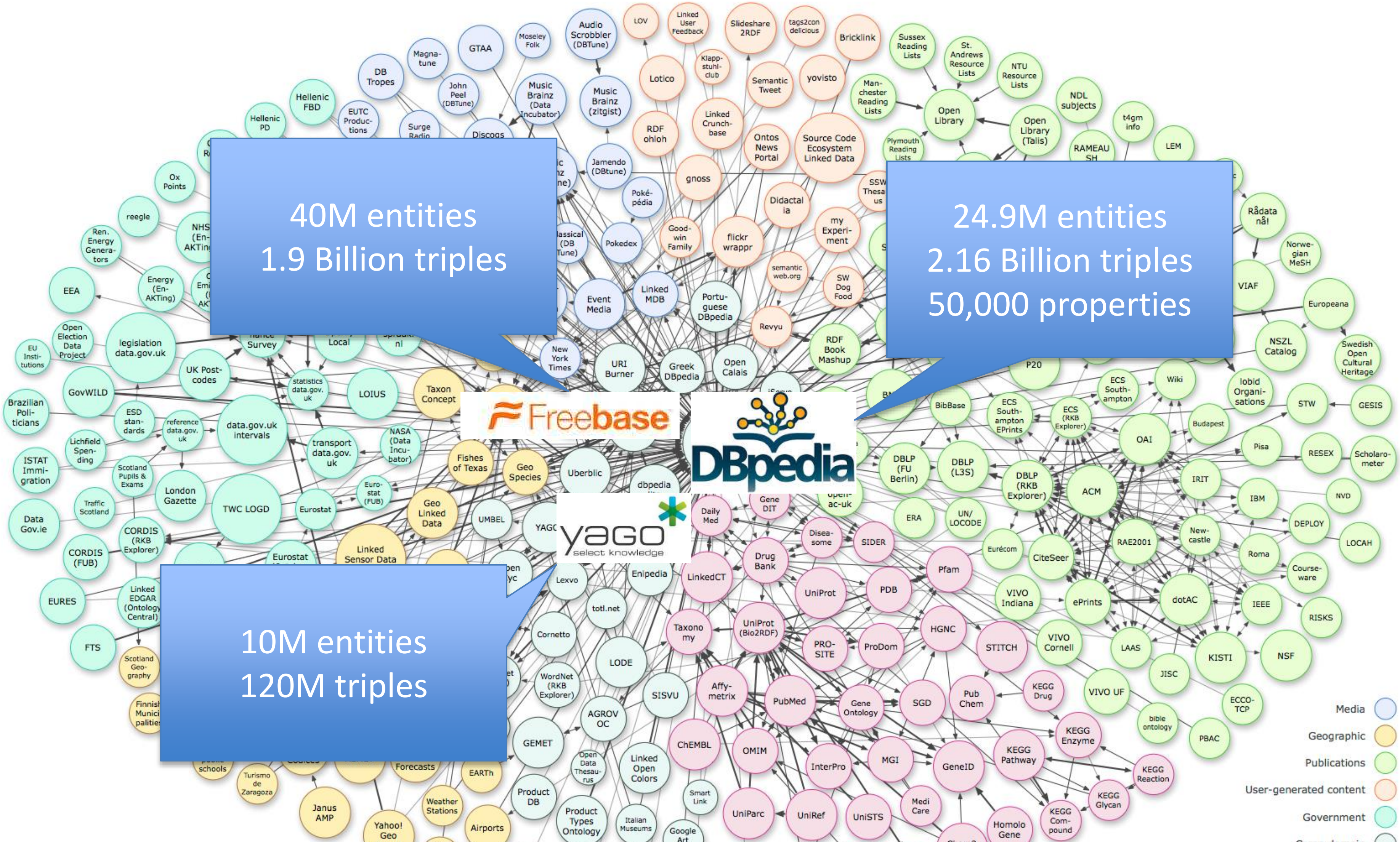


LOD Cloud and the Web of Data



adapted from
Suchanek & Weikum tutorial@SIGMOD 2013

LOD Cloud and the Web of Data



adapted from
Suchanek & Weikum tutorial@SIGMOD 2013

Entities: An Invaluable Asset

Monuments

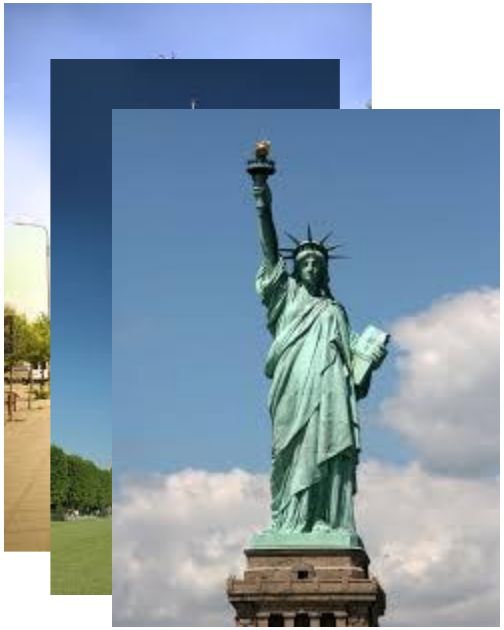
"Entities" is what a large part of our knowledge is about

Entities: An Invaluable Asset

Monuments

"Entities" is what a large part of our knowledge is about

Entities: An Invaluable Asset

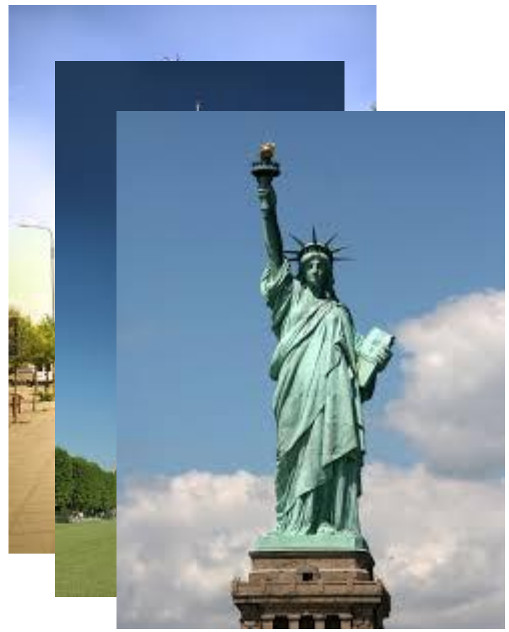


Monuments

"Entities" is what a large part of our knowledge is about

Entities: An Invaluable Asset

Locations

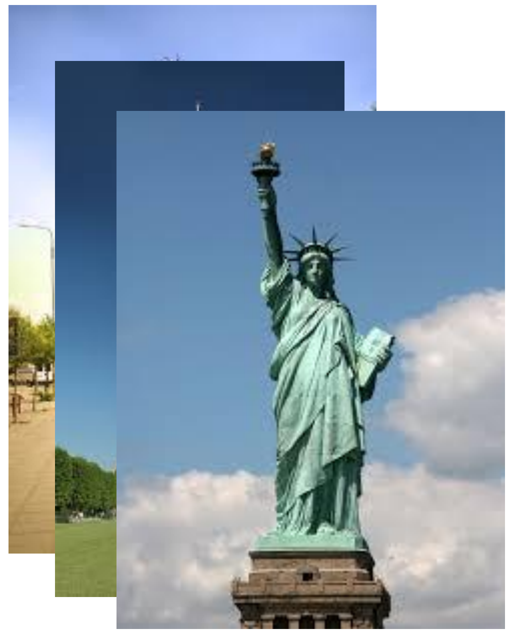


Monuments

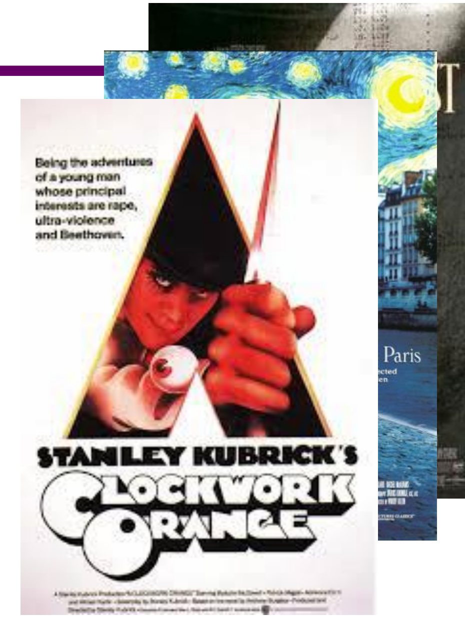
"Entities" is what a large part of our knowledge is about

Entities: An Invaluable Asset

Locations



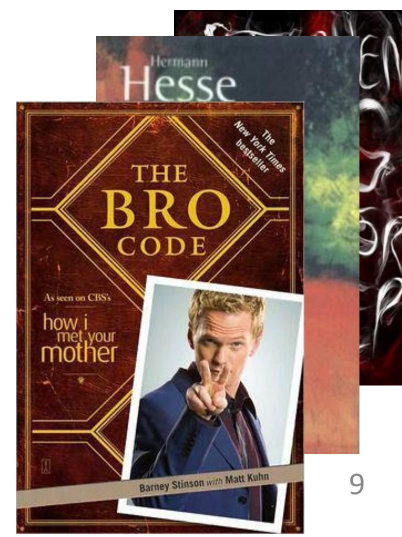
Monuments



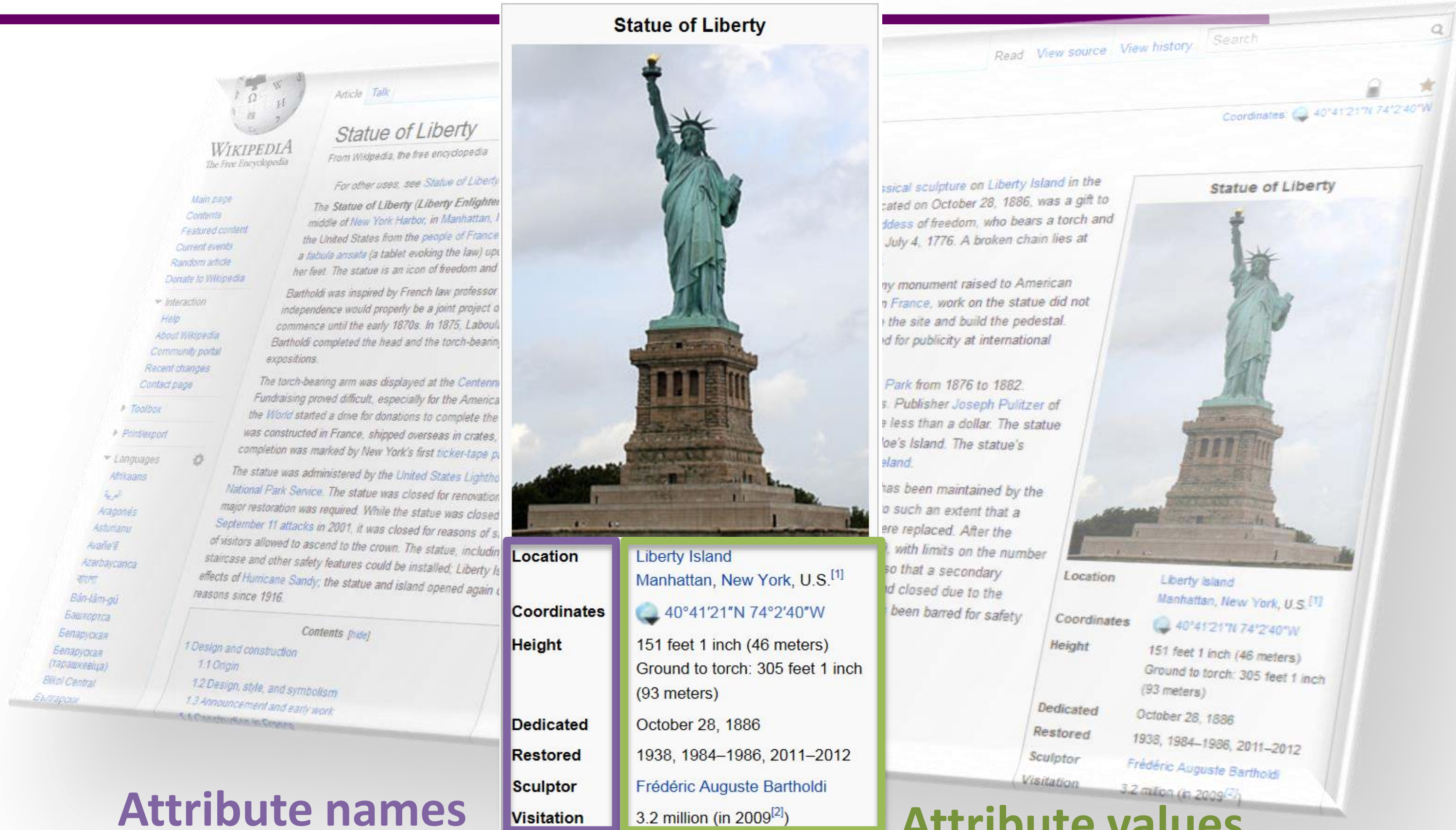
Movies

Persons

Books



Example: General Knowledge Bases



Different Descriptions of the same Entity

DBpedia	dbpedia:Statue_of_Liberty
rdfs:label	Statue of Liberty, Freiheitsstatue, …
dbpprop:location	New York City, New York, U.S., dbpedia:Liberty_Island
dbpprop:sculptor	dbpedia:Frédéric_Auguste_Bartholdi
dcterms:subject	dbpedia_category:1886_sculptures, …
foaf:isPrimaryTopicOf	http://en.wikipedia.org/wiki/Statue_of_Liberty
dbpprop:beginningDate	1886-10-28 (xsd:date)
dbpprop:restored	19381984 (xsd:integer)
dbpprop:visitationNum	3200000 (xsd:integer)
dbpprop:visitationYear	2009 (xsd:integer)
http://www.w3.org/ns/prov#wasDerivedFrom	http://en.wikipedia.org/wiki/Statue_of_Liberty?oldid=494328330

Freebase	fb:m.072p8
fb:art_form	fb:m.06msq (Sculpture)
fb:media	fb:m.025rsfk (Copper)
fb:architect	fb:m.0jph6 (F. Bartholdi), fb:m.036qb (G. Eiffel), fb:m.02wj4z (R. Hunt)
fb:height_meters	93
fb:opened	1886-10-28

yago select knowledge	yago:Statue_of_Liberty
skos:prefLabel	Statue of Liberty
rdf:type	yago:History_museums_in_NY, yago:GeoEntity
yago:hasHeight	46.0248
yago:wasCreatedOnDate	1886-##-##
yago:isLocatedIn	yago:Manhattan, yago:Liberty_Island,
yago:hasWikipediaUrl	http://en.wikipedia.org/wiki/Statue_of_Liberty

Linked Datasets Depend on Vocabularies

DBpedia	dbpedia:Statue_of_Liberty
rdfs:label	Statue of Liberty, Freiheitsstatue, …
dbpprop:location	New York City, New York, U.S., dbpedia:Liberty_Island
dbpprop:sculptor	dbpedia:Frédéric_Auguste_Bartholdi
dcterms:subject	dbpedia_category:1886_sculptures, …
foaf:isPrimaryTopicOf	http://en.wikipedia.org/wiki/Statue_of_Liberty
dbpprop:beginningDate	1886-10-28 (xsd:date)
dbpprop:restored	19381984 (xsd:integer)
dbpprop:visitationNum	3200000 (xsd:integer)
dbpprop:visitationYear	2009 (xsd:integer)
http://www.w3.org/ns/prov#wasDerivedFrom	http://en.wikipedia.org/wiki/Statue_of_Liberty?oldid=494328330

Freebase	fb:m.072p8
fb:art_form	fb:m.06msq (Sculpture)
fb:media	fb:m.025rsfk (Copper)
fb:architect	fb:m.0jph6 (F. Bartholdi), fb:m.036qb (G. Eiffel), fb:m.02wj4z (R. Hunt)
fb:height_meters	93
fb:opened	1886-10-28

yago select knowledge	yago:Statue_of_Liberty
skos:prefLabel	Statue of Liberty
rdf:type	yago:History_museums_in_NY, yago:GeoEntity
yago:hasHeight	46.0248
yago:wasCreatedOnDate	1886-##-##
yago:isLocatedIn	yago:Manhattan, yago:Liberty_Island,
yago:hasWikipediaUrl	http://en.wikipedia.org/wiki/Statue_of_Liberty

Linked Datasets Have Varying Quality

DBpedia	dbpedia:Statue_of_Liberty
rdfs:label	Statue of Liberty, Freiheitsstatue, …
dbpprop:location	New York City, New York, U.S., dbpedia:Liberty_Island
dbpprop:sculptor	dbpedia:Frédéric_Auguste_Bartholdi
dcterms:subject	dbpedia_category:1886_sculptures, …
foaf:isPrimaryTopicOf	http://en.wikipedia.org/wiki/Statue_of_Liberty
dbpprop:beginningDate	1886-10-28 (xsd:date)
dbpprop:restored	19381984 (xsd:integer)
dbpprop:visitationNum	3200000 (xsd:integer)
dbpprop:visitationYear	2009 (xsd:integer)
http://www.w3.org/ns/prov#wasDerivedFrom	http://en.wikipedia.org/wiki/Statue_of_Liberty?oldid=494328330

Freebase	fb:m.072p8
fb:art_form	fb:m.06msq (Sculpture)
fb:media	fb:m.025rsfk (Copper)
fb:architect	fb:m.0jph6 (F. Bartholdi), fb:m.036qb (G. Eiffel), fb:m.02wj4z (R. Hunt)
fb:height_meters	93
fb:opened	1886-10-28

yago select knowledge	yago:Statue_of_Liberty
skos:prefLabel	Statue of Liberty
rdf:type	yago:History_museums_in_NY, yago:GeoEntity
yago:hasHeight	46.0248
yago:wasCreatedOnDate	1886-##-##
yago:isLocatedIn	yago:Manhattan, yago:Liberty_Island,
yago:hasWikipediaUrl	http://en.wikipedia.org/wiki/Statue_of_Liberty

The Problem Entity Resolution

We need to identify that all descriptions refer to the same real-world object

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

A prerequisite to several applications:

- Enable semantic search in terms of entities & relations (*on top of the web of text*)
- Interlink entity descriptions in autonomous sources (*strengthen the web of data*)
- Support deep reasoning using related ontologies (*create the web of knowledge*)

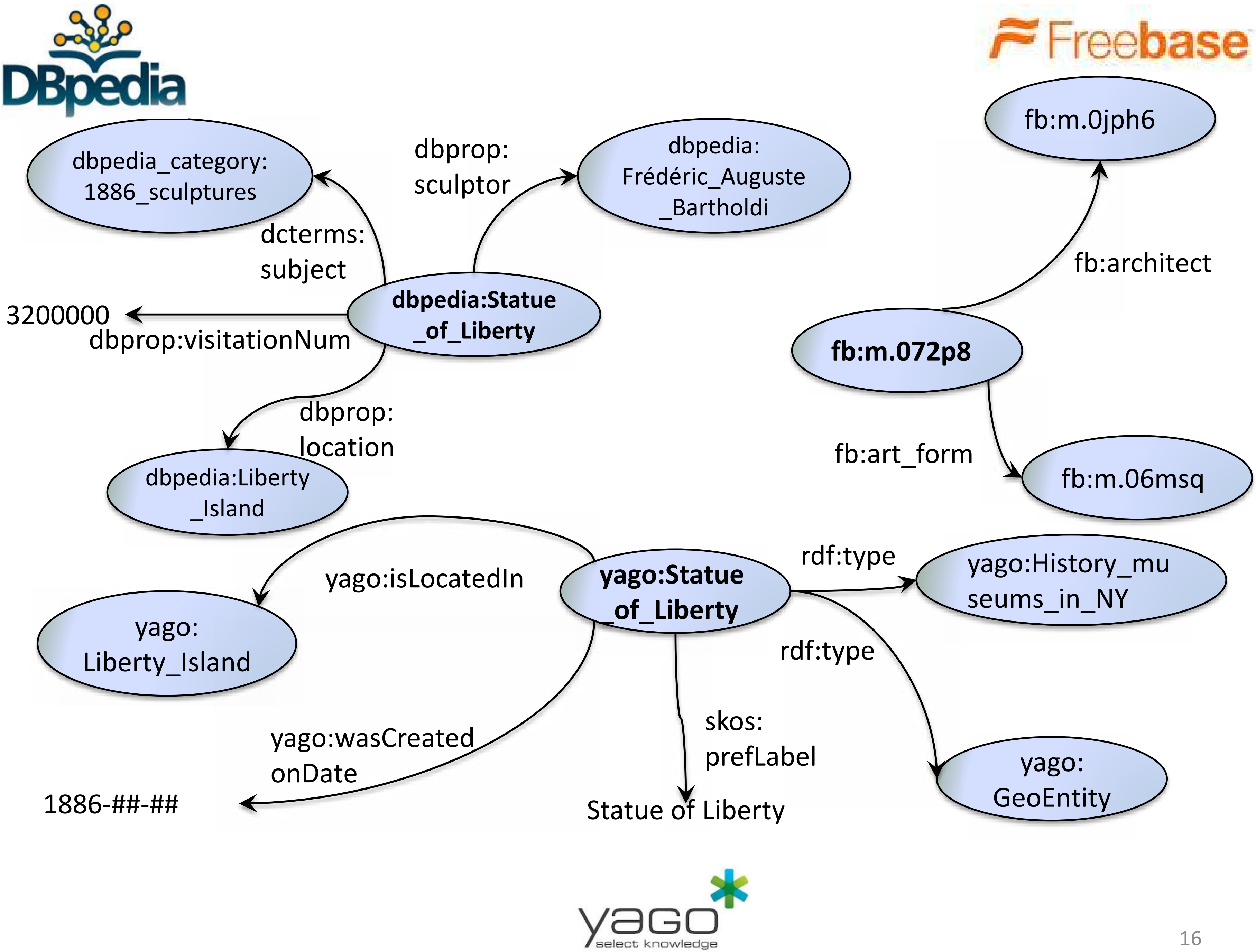
Entity Collections and Entity Resolution Types

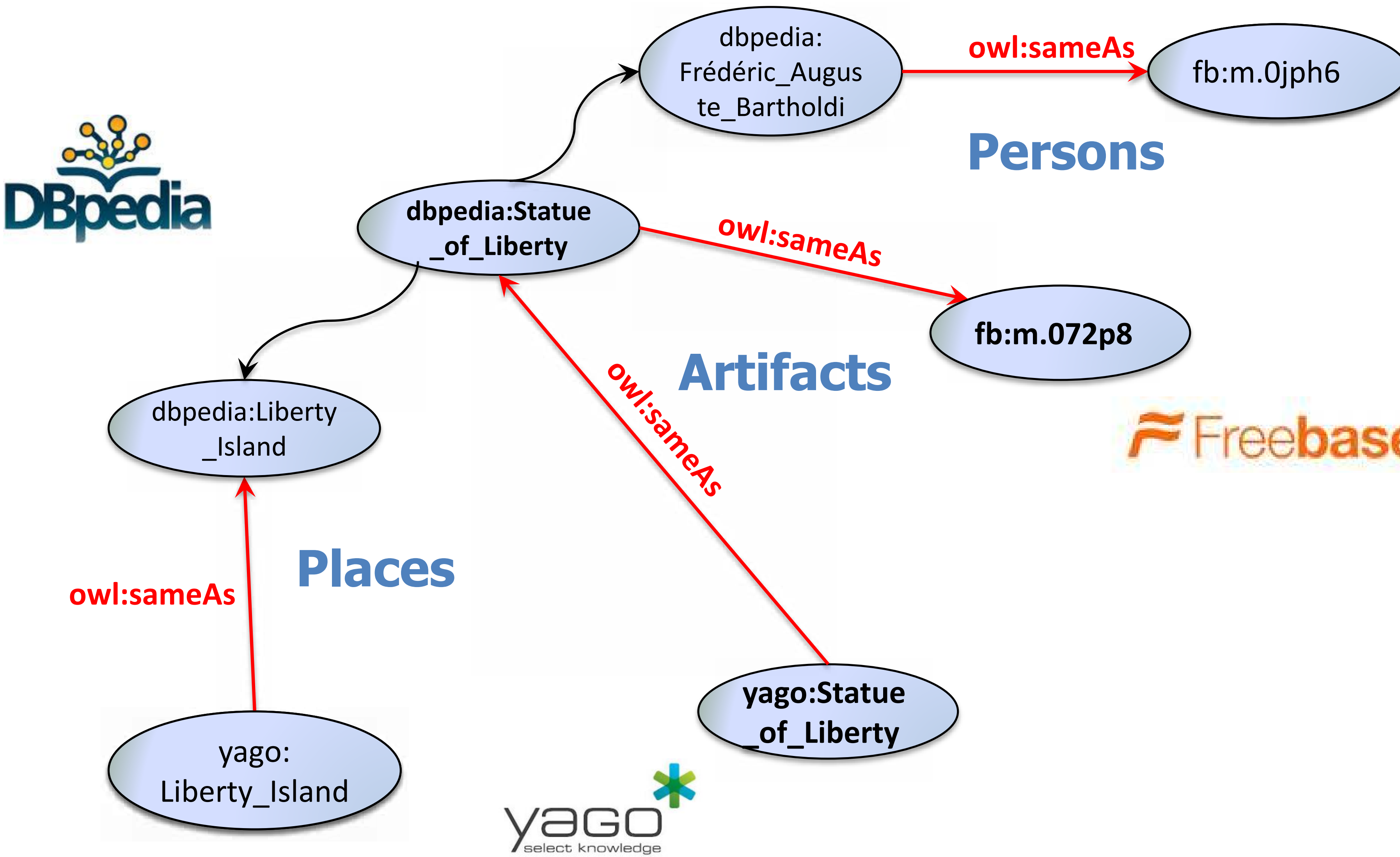
Two kinds of entity collections as input:

- Clean: duplicate-free
- Dirty: contains duplicate entity descriptions

An entity resolution task can be:

- Clean-Clean Entity Resolution: Given two clean, but overlapping entity collections, identify the common entity descriptions
  - a.k.a. *record linkage* in databases
- Dirty-Clean Entity Resolution
- Dirty Entity Resolution: Identify unique entity descriptions contained in one dirty entity collection
  - a.k.a. *deduplication* in databases


DBpedia
Freebase
dbpedia_category:
1886_sculptures
dcterms:
subject
dbprop:
sculptor
dbpedia:
Frédéric_Auguste
_Bartholdi
dbpedia:Statue
_of_Liberty
3200000
dbprop:visitationNum
dbprop:
location
dbpedia:Liberty
_Island
fb:m.0jph6
fb:architect
fb:m.072p8
fb:art_form
fb:m.06msq
yago:isLocatedIn
yago:
Liberty_Island
yago:Statue
_of_Liberty
rdf:type
yago:History_mu
seums_in_NY
rdf:type
yago:
GeoEntity
skos:
prefLabel
Statue of Liberty
yago:wasCreated
onDate
1886-##-##
yago
select knowledge



⇒Need to infer also other kind of relationships than “equivalence”

What Makes Entity Resolution Difficult for the Web of Data

- Linked Data are inherently semi-structured
  - Several semantic types could be employed (see rdf:type properties in Yago), resulting to quite different structures even for entity descriptions of the same type (persons, places, …)

=> Deal with loosely structured entities

- Linked Data heavily rely on various vocabularies
  - 366 distinct vocabulary spaces in the LOD cloud (http://lov.okfn.org/dataset/lov/)
  - DBPedia 3.4: 50,000 attribute names

=> Need for cross-domain techniques

- Linked Data are Big (semi-structured) Data
  - LOD cloud: 60 billion RDF triples
  - DBPedia 3.9: 2.46 billion triples, 24.9 million entity descriptions
  - Freebase: 1.9 billion triples, 40 million entity descriptions
  - Yago: >10 million entities, >120 million triples

=> Call for efficient parallel techniques

Problem Statement

Entity Description

Each description is expressed as a set of attribute-value pairs

An entity description $e_i \in E$ is defined as: $e_i = \{(a_{ij}, v_{ij}) \mid a_{ij} \in N, v_{ij} \in V\}$

N: a set of attribute names

V: a set of values

E: a set of entity descriptions

We use a generic definition for entity descriptions to cover different data models

Structural type of e_i: the set of attributes along with their domains in e_i

- In the Web of data, the descriptions even of the same entities do not always conform to the same structural type

Entity Description Examples

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

Entity Resolution – Formal Definition

Entity resolution: The problem of identifying descriptions of the same entity within one or across multiple data sources wrt. a match function

Formally:

$E = \{e_1, \ldots, e_m\}$ is a set of entity descriptions

$M : E \times E \rightarrow \{true, false\}$ is a match function

An entity resolution of E is a partition $P = \{p_1, \ldots, p_n\}$ of E, such that:

1. $\forall e_i, e_j \in E : M(e_i, e_j) = true, \exists p_k \in P : e_i, e_j \in p_k$
2. $\forall p_k \in P, \forall e_i, e_j \in p_k, M(e_i, e_j) = true$

each partition contains only matching descriptions

all the matching descriptions are in the same partition

Entity Resolution - Example

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

e1

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

e4

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

e2

about	**Lady liberty**
architect	Eiffel
location	NY

e3

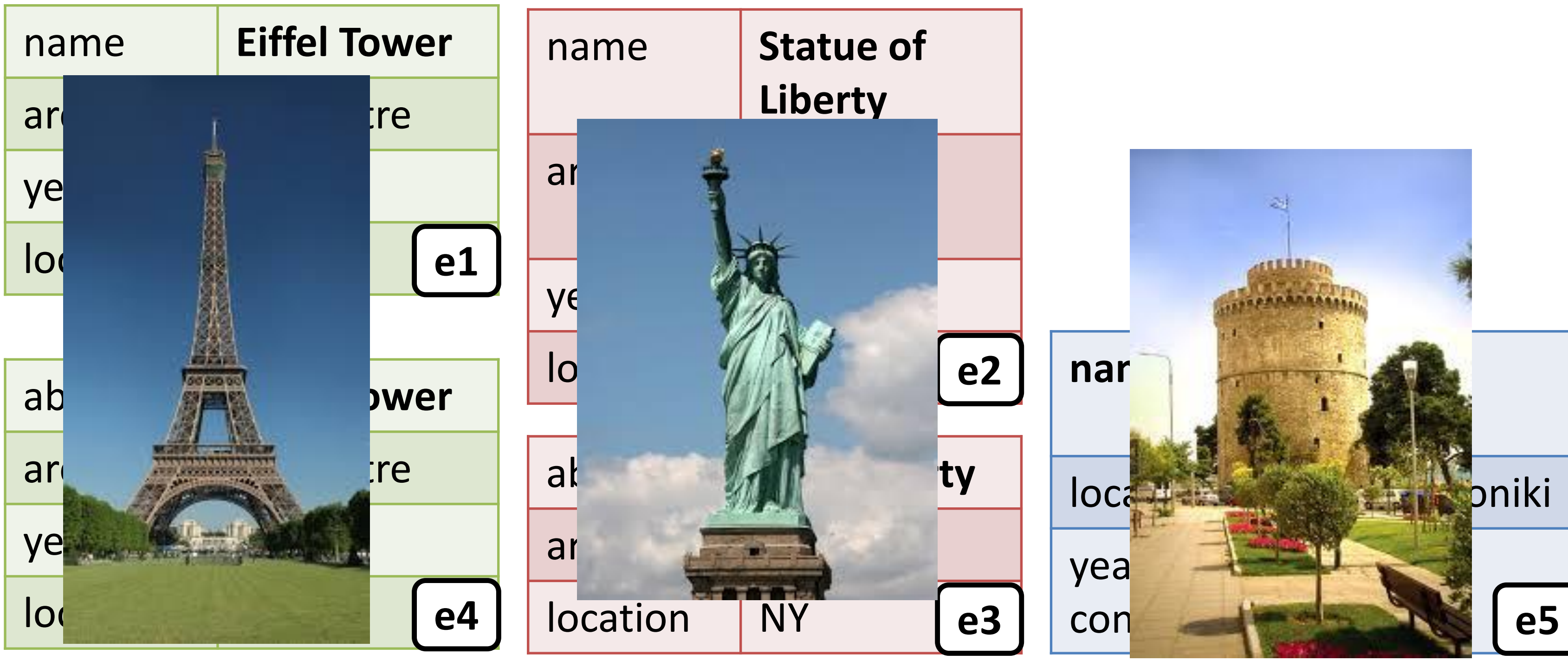
name	**White Tower**
location	Thessaloniki
year-constructed	1450

e5

Assume as input of entity resolution, the set $E = \{e_1, e_2, e_3, e_4, e_5\}$

- A possible output $P = \{\{e_1, e_4\}, \{e_2, e_3\}, \{e_5\}\}$ indicates that:

Entity Resolution - Example



Assume as input of entity resolution, the set E = $\{e_1, e_2, e_3, e_4, e_5\}$

- A possible output P = $\{\{e_1, e_4\}, \{e_2, e_3\}, \{e_5\}\}$ indicates that:
  - e_1, e_4 refer to the same real-world object, the Eiffel Tower
  - e_2, e_3 represent a different object, the Statue of Liberty
  - e_5 represents a third object, the White Tower

Entity Resolution - Match

Matches: Sets of entity descriptions that refer to the same real-world entity

Intuitively:

- Matching entity descriptions are placed in the same subset of P
- All the descriptions of the same subset of P match

A match function maps each pair of entity descriptions (e_i, e_j) to {true, false}

- $M(e_i, e_j)$ = true => e_i, e_j are matching descriptions
- $M(e_i, e_j)$ = false => e_i, e_j are non-matches

Entity Resolution - Similarity

Typically, the match function is expressed wrt. a similarity measure sim

– *sim* counts how close two entity descriptions are to each other

Given a similarity threshold t:

– $M(e_i, e_j)$ = true, if $sim(e_i, e_j) \geq t$

– $M(e_i, e_j)$ = false, if $sim(e_i, e_j) < t$

Similarity of Entity Descriptions

How can we identify that two entity descriptions refer to the same entity?

Similarity of Entity Descriptions

How can we identify that two entity descriptions refer to the same entity?

- If they are identical, then we assume they match (exact match function)

E.g.

name	Eiffel Tower	
architect	Sauvestre	
year	1889	
location	Paris	**e1**

name	Eiffel Tower	
architect	Sauvestre	
year	1889	
location	Paris	**e2**

Similarity of Entity Descriptions

How can we identify that two entity descriptions refer to the same entity?

- If they are identical, then we assume they match (exact match function)
  - Even this assumption could be false!

E.g.

first	John	
last	Doe	
born	1980	
location	UK	**e1**

first	John	
last	Doe	
born	1980	
location	UK	**e2**

… could describe namesakes, born in the same country and year

Similarity of Entity Descriptions

How can we identify that two entity descriptions refer to the same entity?

- What if they are not identical, but it looks like they match?
  - e.g.

about	Gustave Eiffel	**e1**

name	G. Eiffel	**e2**

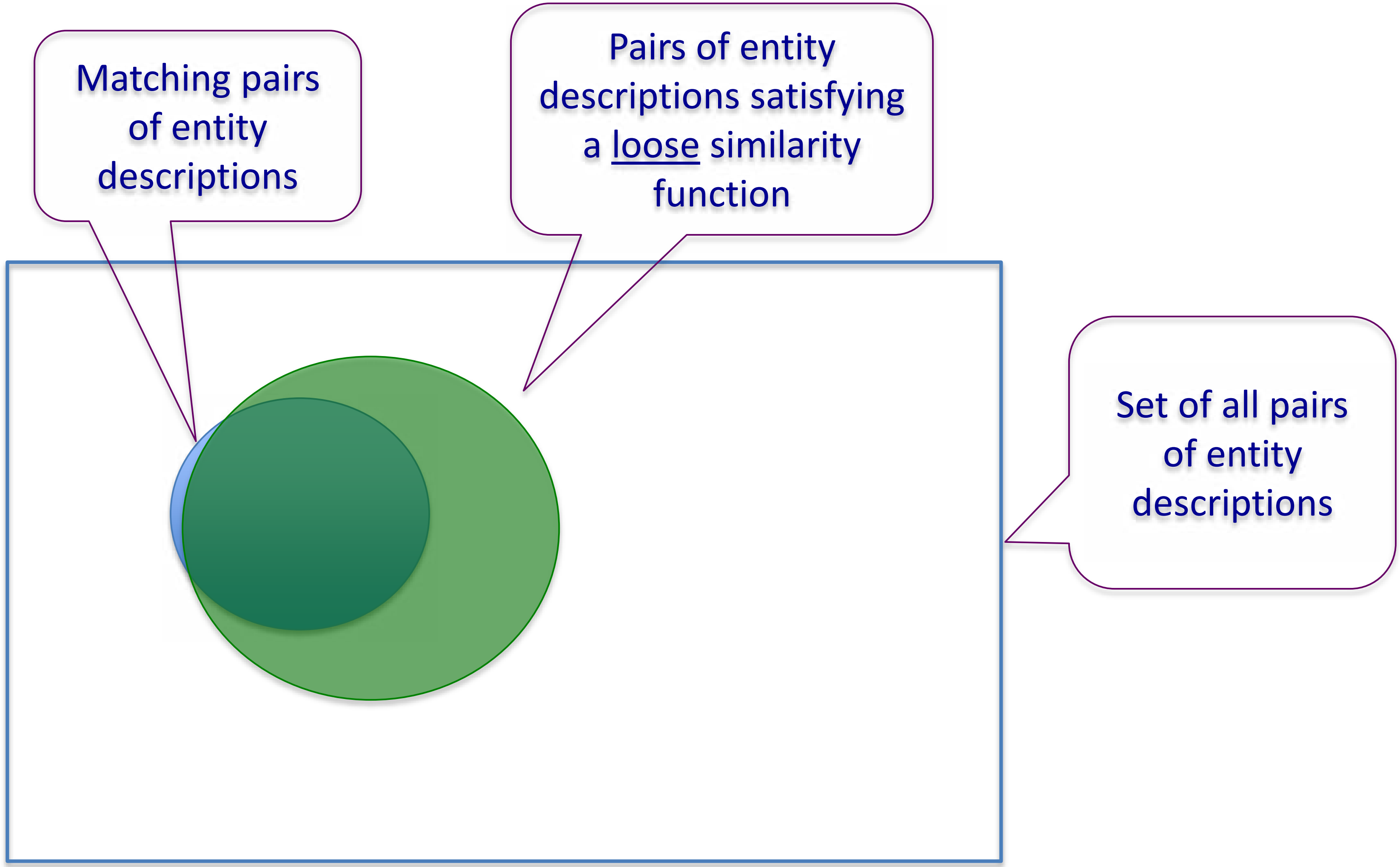
Exact match is rather impractical for entity resolution in the Web of data

- Too strict for a highly heterogeneous information space

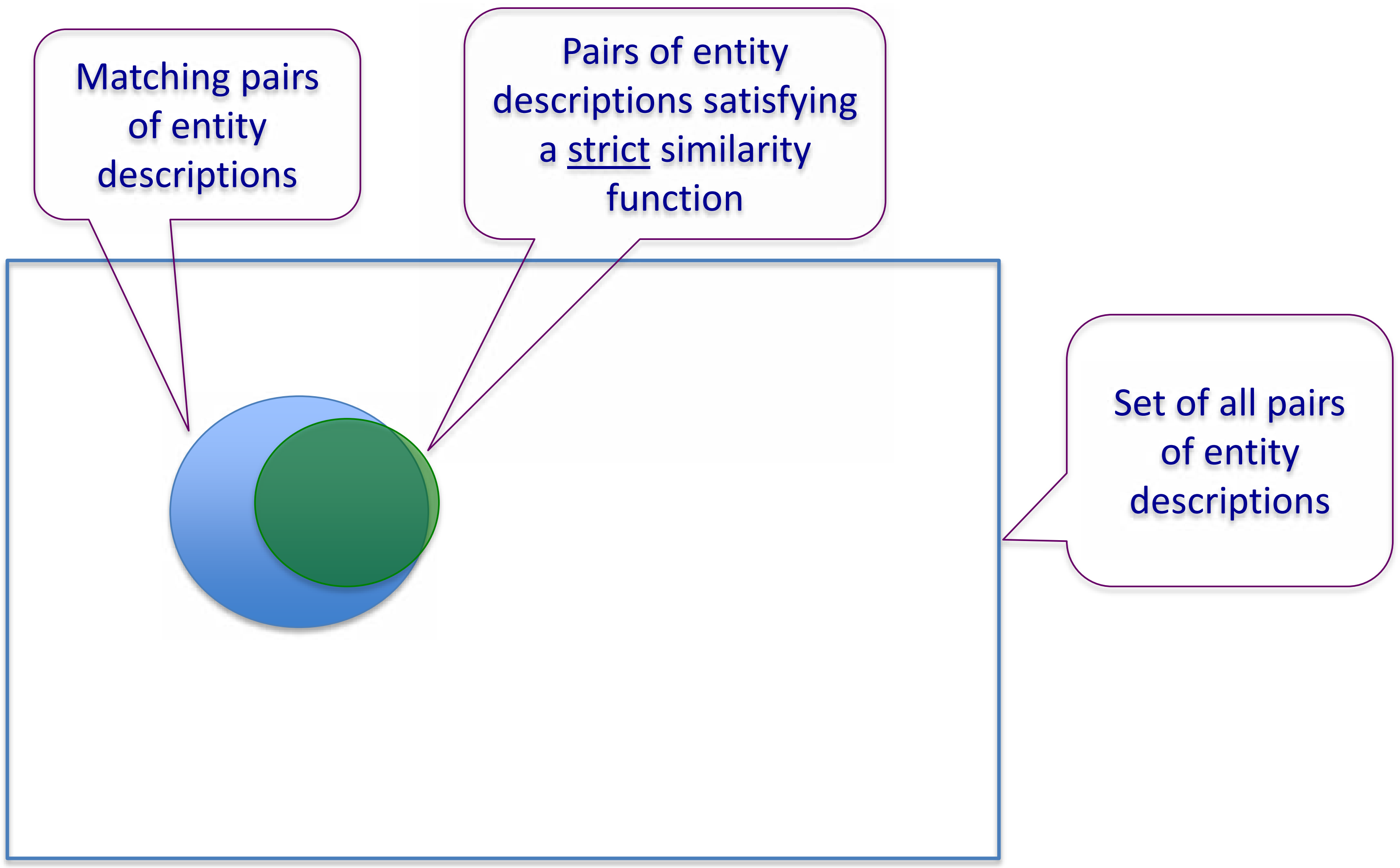
A more loose similarity measure could identify more matches, but…

- Which similarity measure is that?
- What should it compare? Values/Structure/Neighbors?
- It might be too loose and return many false matches too!

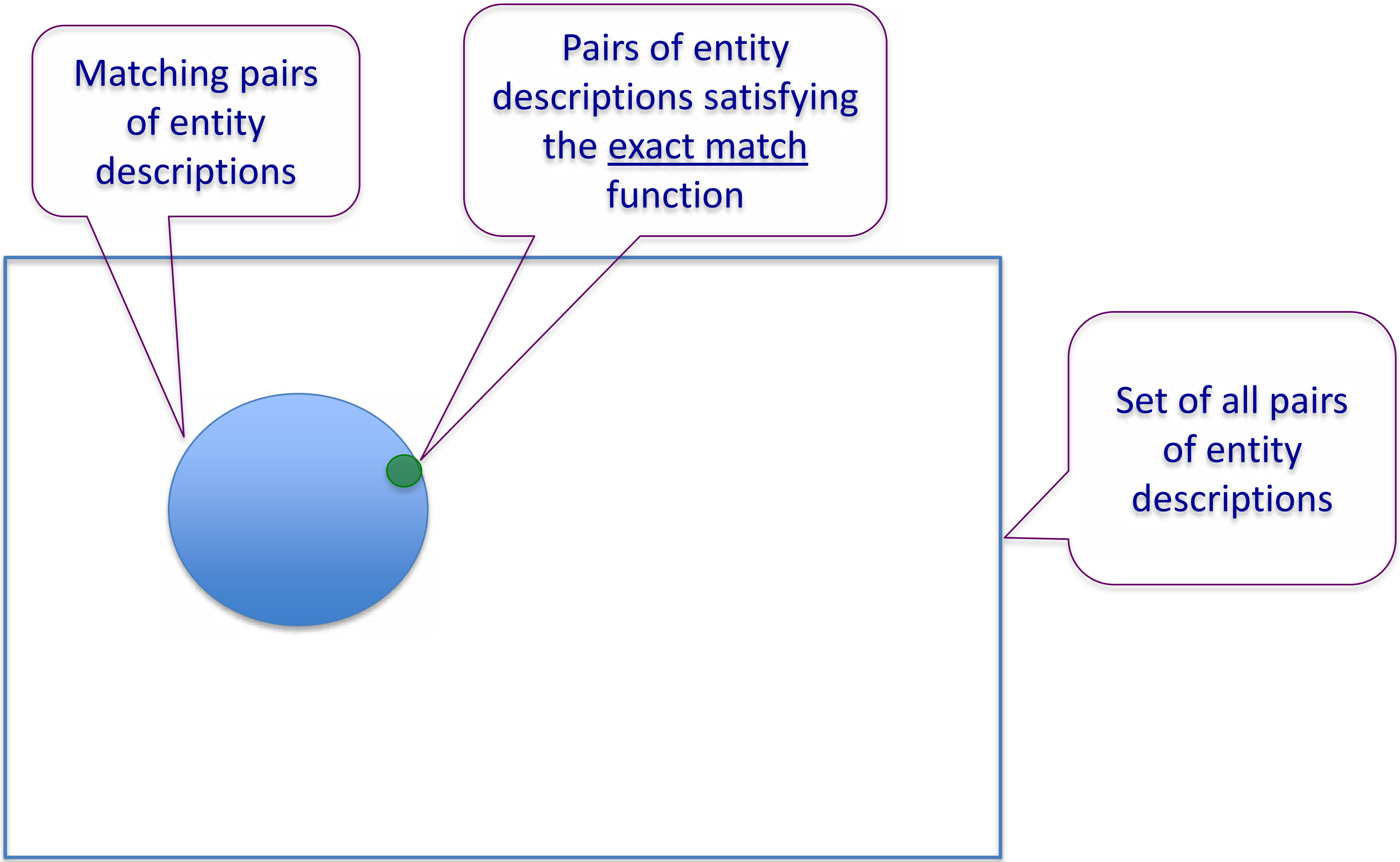
The Role of Similarity Functions – Loose Function



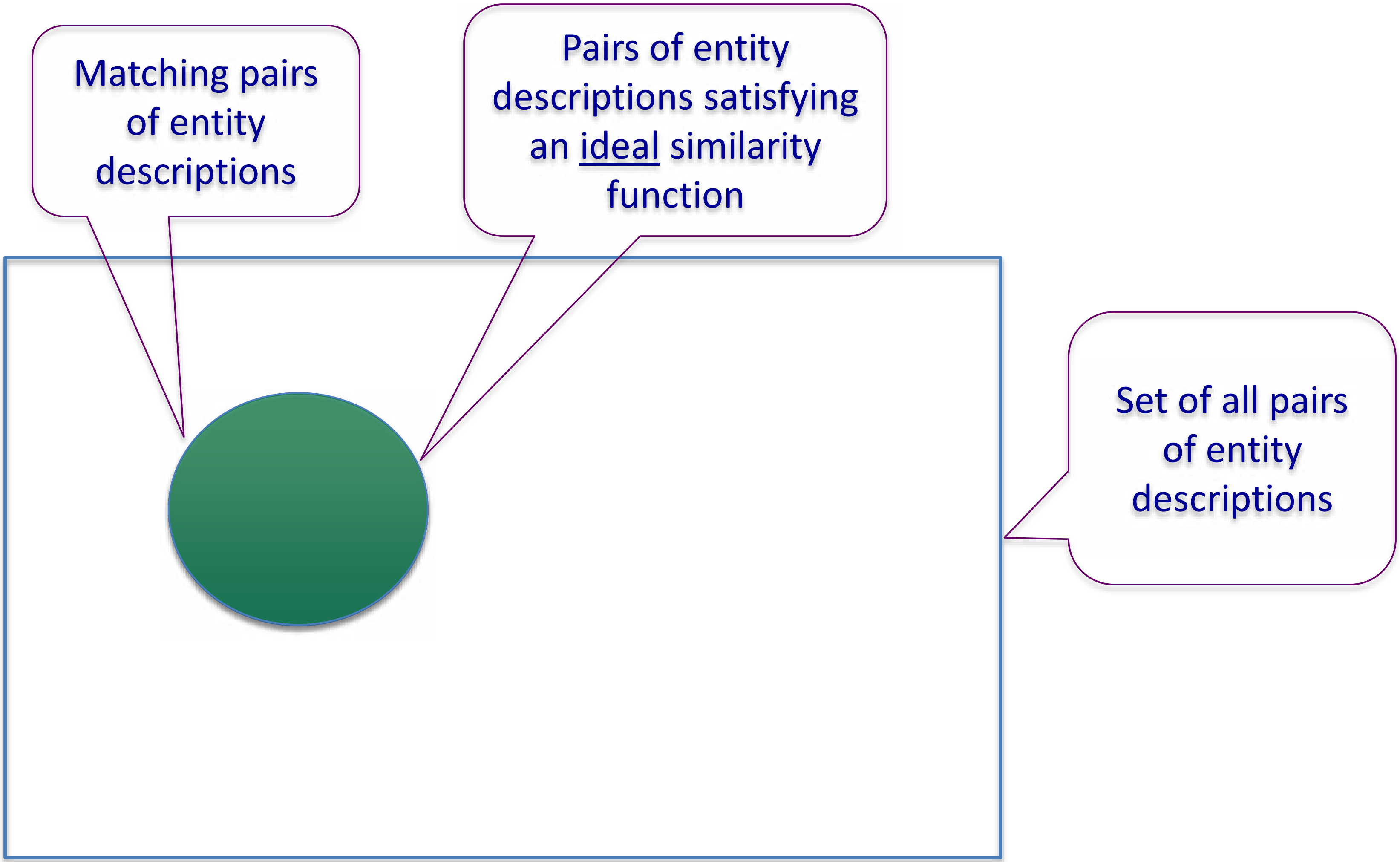
The Role of Similarity Functions – Strict Function



The Role of Similarity Functions – Exact Match



The Role of Similarity Functions – Ideal Case



Using Relationships

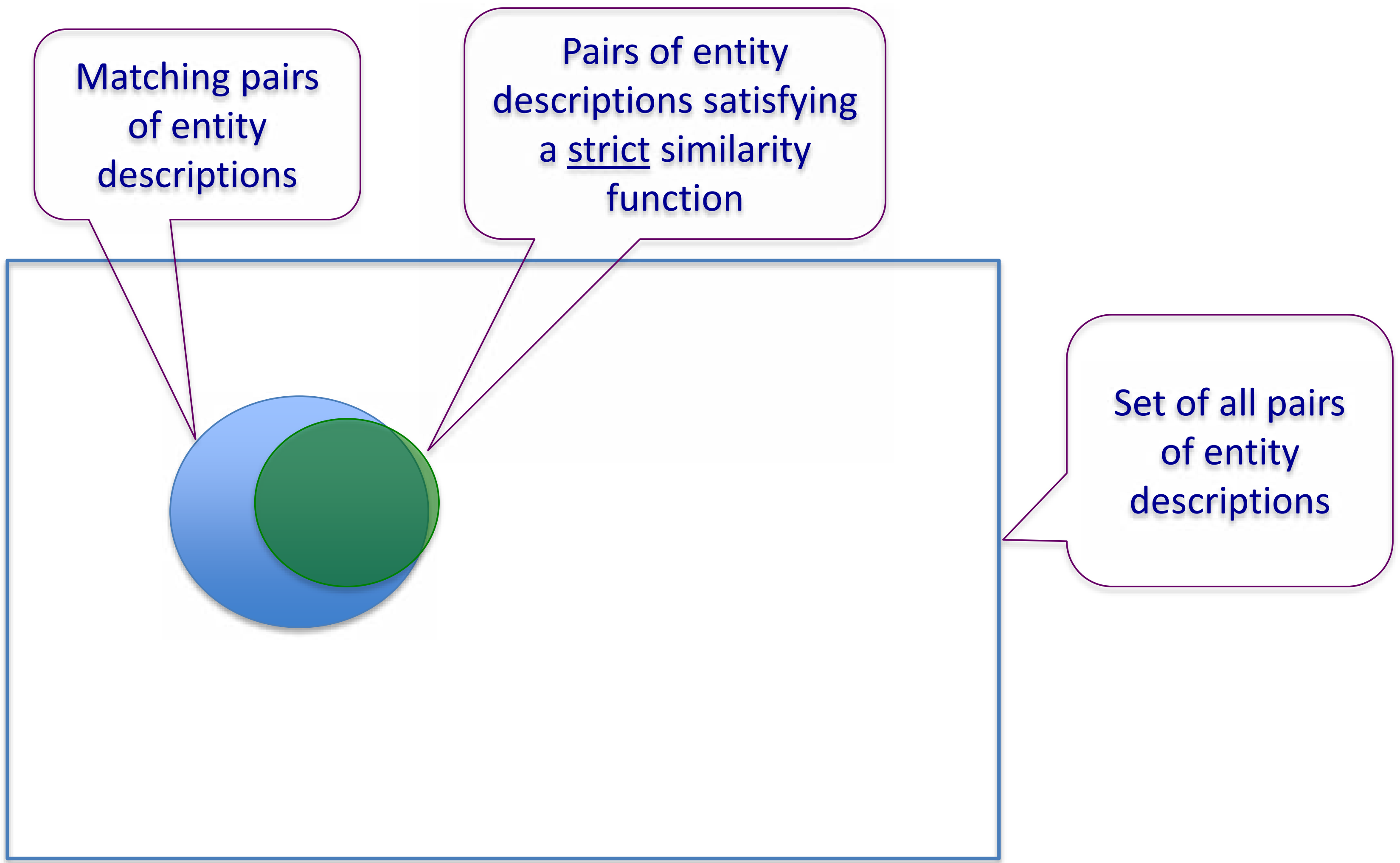
- Transitivity: If (A,B) are matches and (B, C) are matches, then (B,C) are also matches

- Duplicate dependency: If entities Author1 and Author2 are matches, then related entities Publication1 and Publication2 are more likely to be matches than before the matching of Author1 and Author2

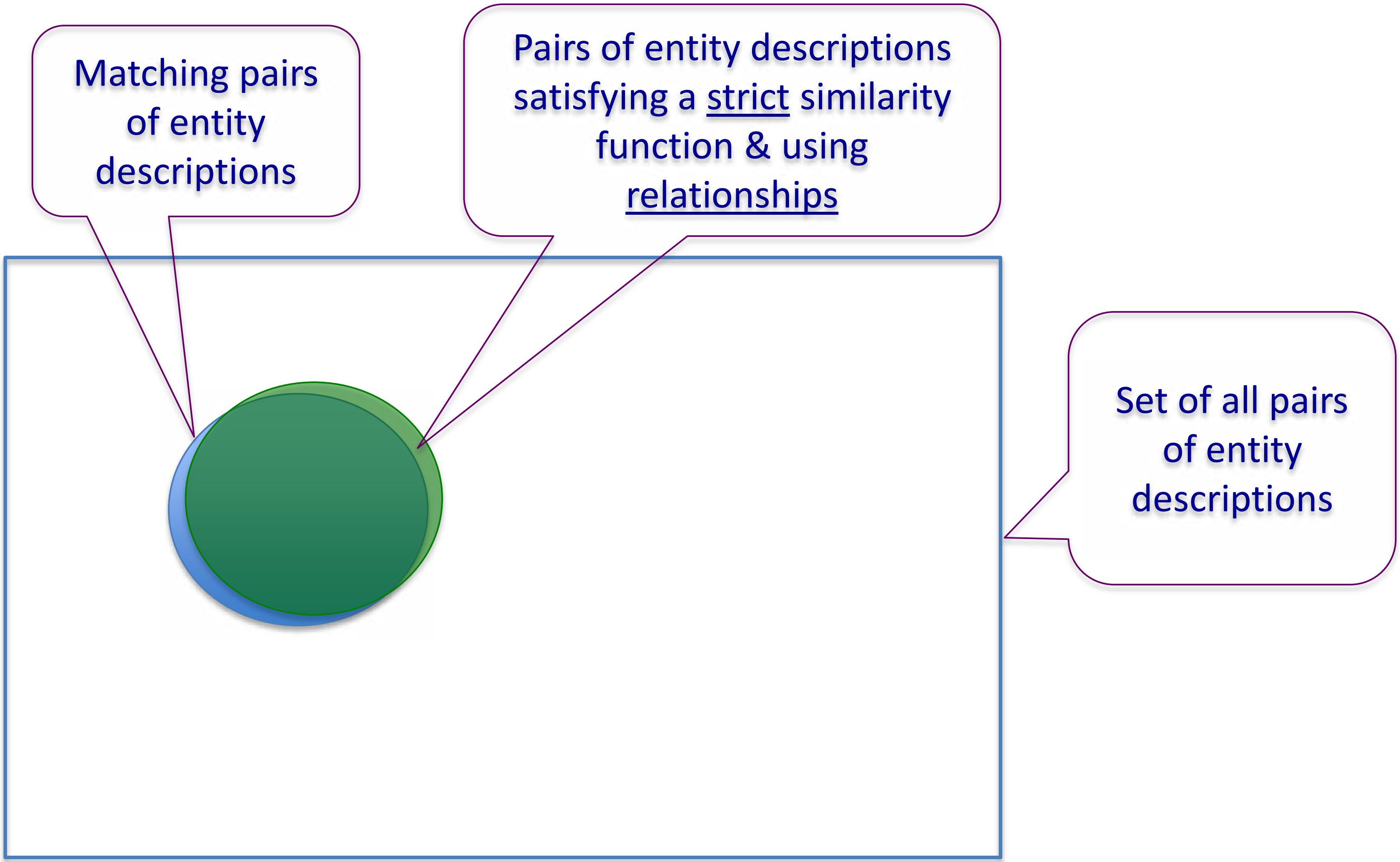
- Merge dependency: Once a matching pair has been identified, the merged entity descriptions create a new description that should be compared to the remaining ones

Using these relationships lead to identifying more matches

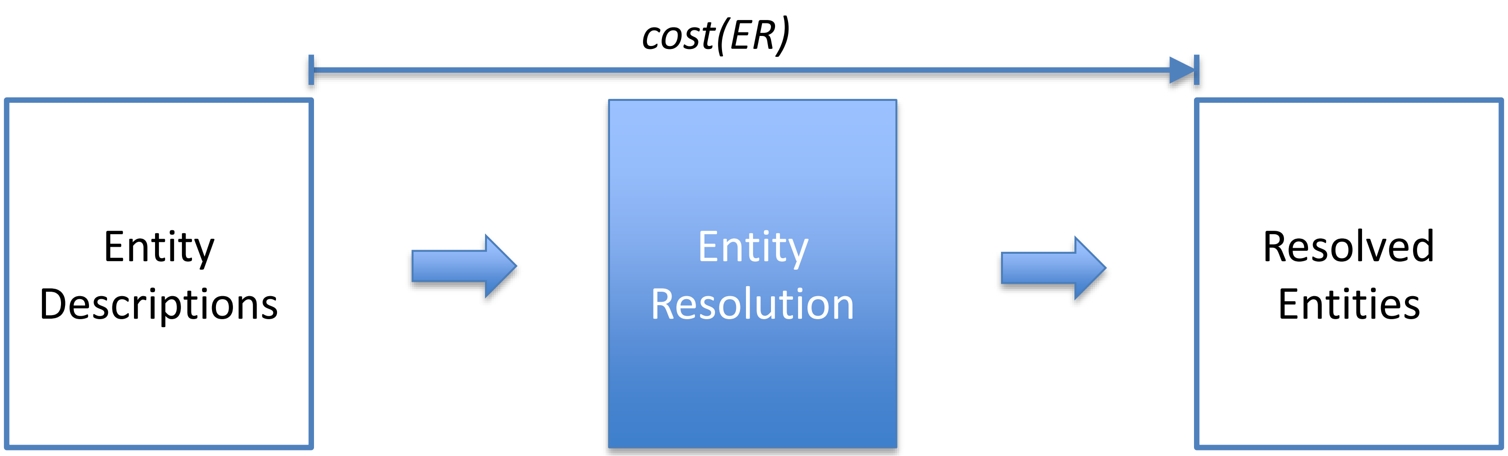
Impact of Using Relationships



Impact of Using Relationships



Entity Resolution Workflow

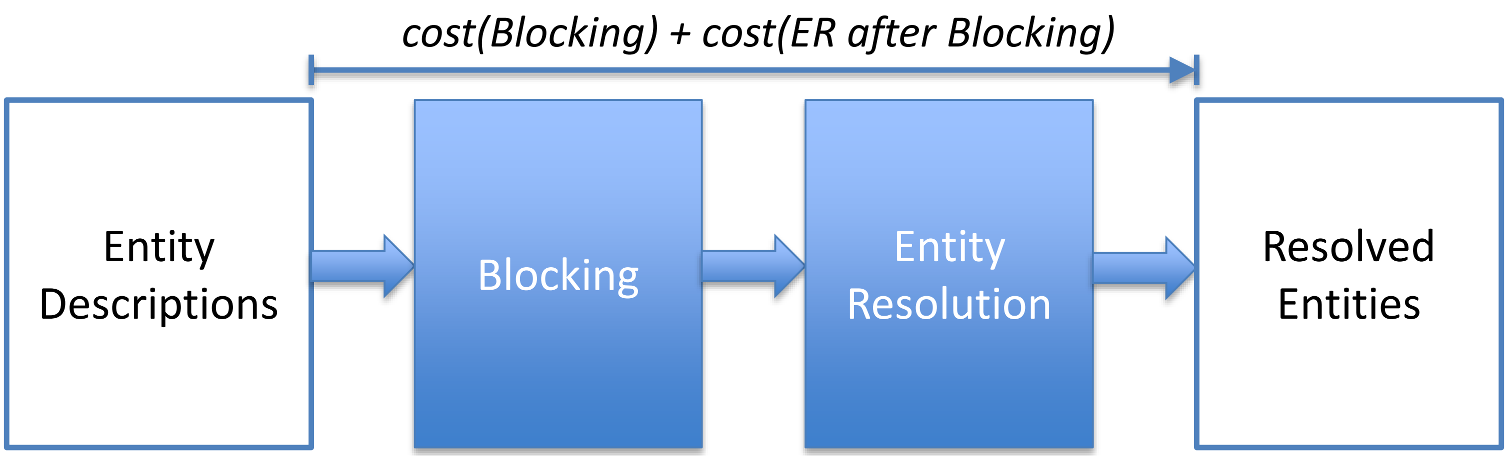


__This is a global optimization problem!__

Good balance between:

- Number of identified matching descriptions
- Number of generated comparisons

Entity Resolution Workflow



A preprocessing step to group together descriptions close to each other - *Fewer comparisons*

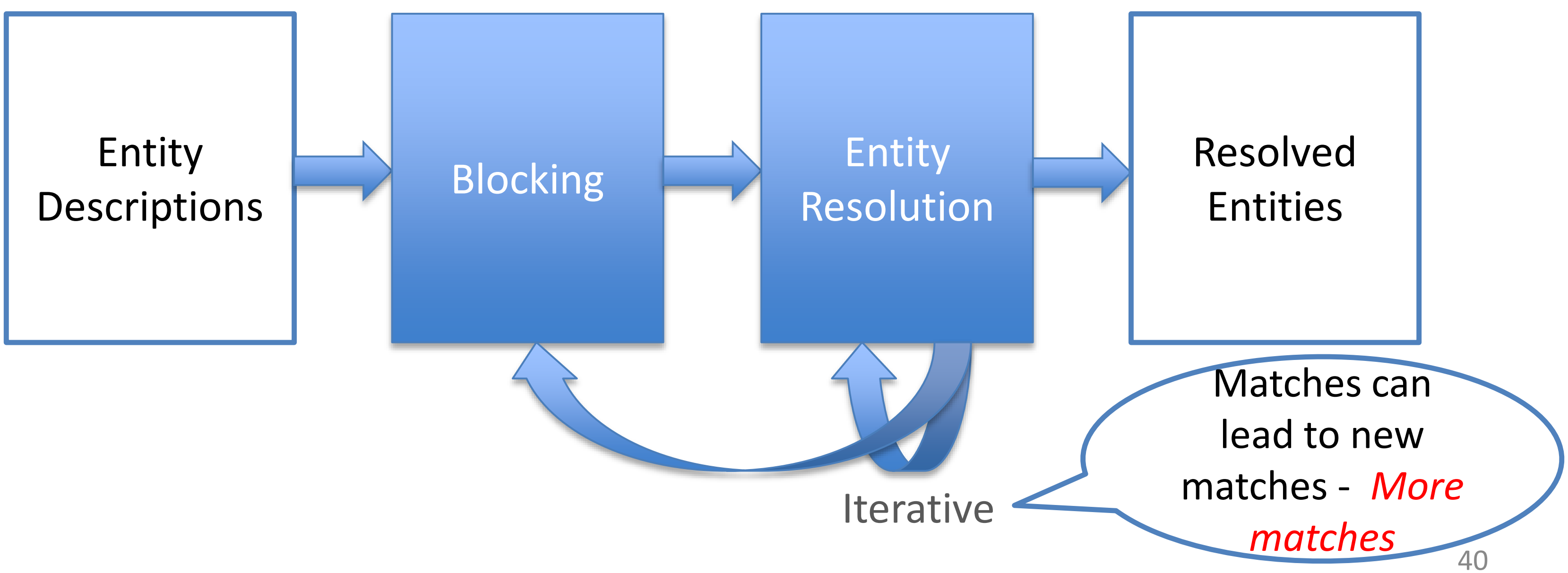
- *cost(ER after Blocking) < cost(ER)*

benefit(Blocking) = cost(ER) – cost(ER after Blocking)

- *cost(Blocking) + cost(ER after Blocking) < cost (ER)*

cost(Blocking) < benefit(Blocking) ???

Entity Resolution Workflow



Blocking Approaches

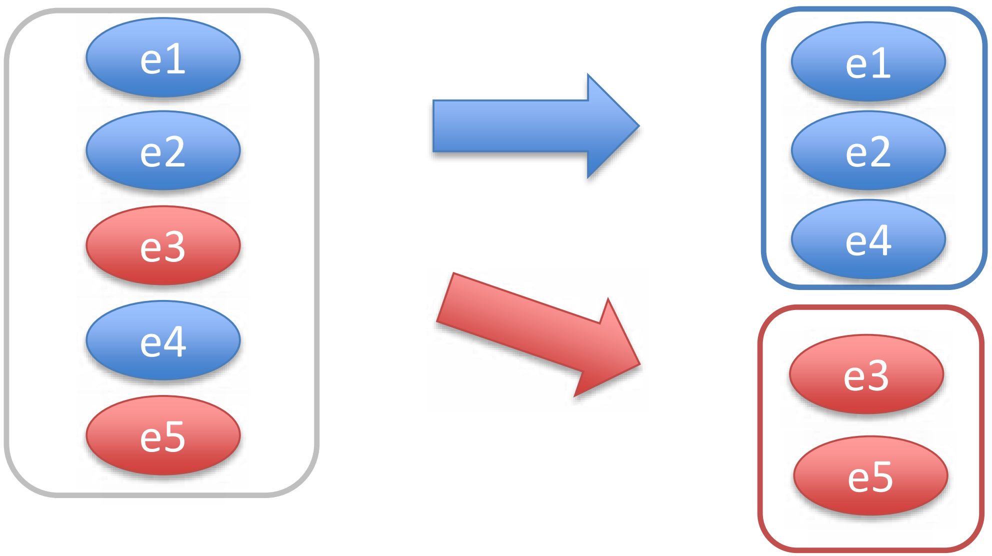
Blocking

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

Sorted Neighborhood Method [Hernandez & Stolfo 1995]

The idea

1. Create key
   - Creates a key value based on relevant attribute values
2. Sort
   - Sort tuples in lexicographical order of their generated keys
3. Merge
   - Slide a window (of fixed size *w*) over the sorted data
   - Limit to comparisons of tuple pairs falling in the same window

Sorted Neighborhood Method

ID	Title	Year	Genre
17	Mask of Zorro	1998	Adventure
18	Addams Family	1991	Comedy
25	Rush Hour	1998	Comedy
31	Matrix	1999	Sci-Fi
52	Return of Dschafar	1994	Children
113	Adams Family	1991	Comedie
207	Return of Djaffar	1995	Children

(1) create key

ID	Key
17	MSKAD98
18	DDMCO91
25	RSHCO98
31	MTRSC99
52	RTRCH94
113	DMSCO91
207	RTRCH95

(2) sort

ID	Key
18	DDMCO91
113	DMSCO91
17	MSKAD98
31	MTRSC99
25	RSHCO98
52	RTRCH94
207	RTRCH95

(3) merge

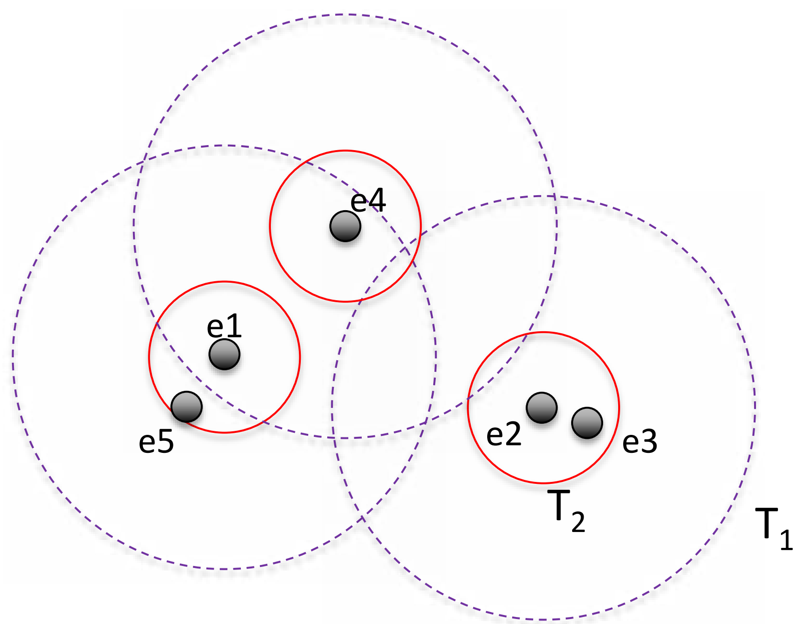
ID	Key
18	DDMCO91
113	DMSCO91
17	MSKAD98
31	MTRSC99
25	RSHCO98
52	RTRCH94
207	RTRCH95

compare(18,113) → duplicates

compare(52,207) → duplicates

Canopy Clustering [McCallum et al. 2000]

1. Pick a random entity description e_i from E
2. Create, for e_i, a new canopy C_{ei}
   Add to C_{ei} the descriptions e_j, s.t. $d(e_i, e_j) < T_1$
3. Remove all descriptions e_j from E, s.t. $d(e_i, e_j) < T_2$
4. Return to Step 1, if E is not empty



Generated Blocks:

e1	e4	e2
e_1, e_4, e_5	e_1, e_4	e_2, e_3

What is the intuition behind thresholds T_1, T_2?

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 entity descriptions 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)

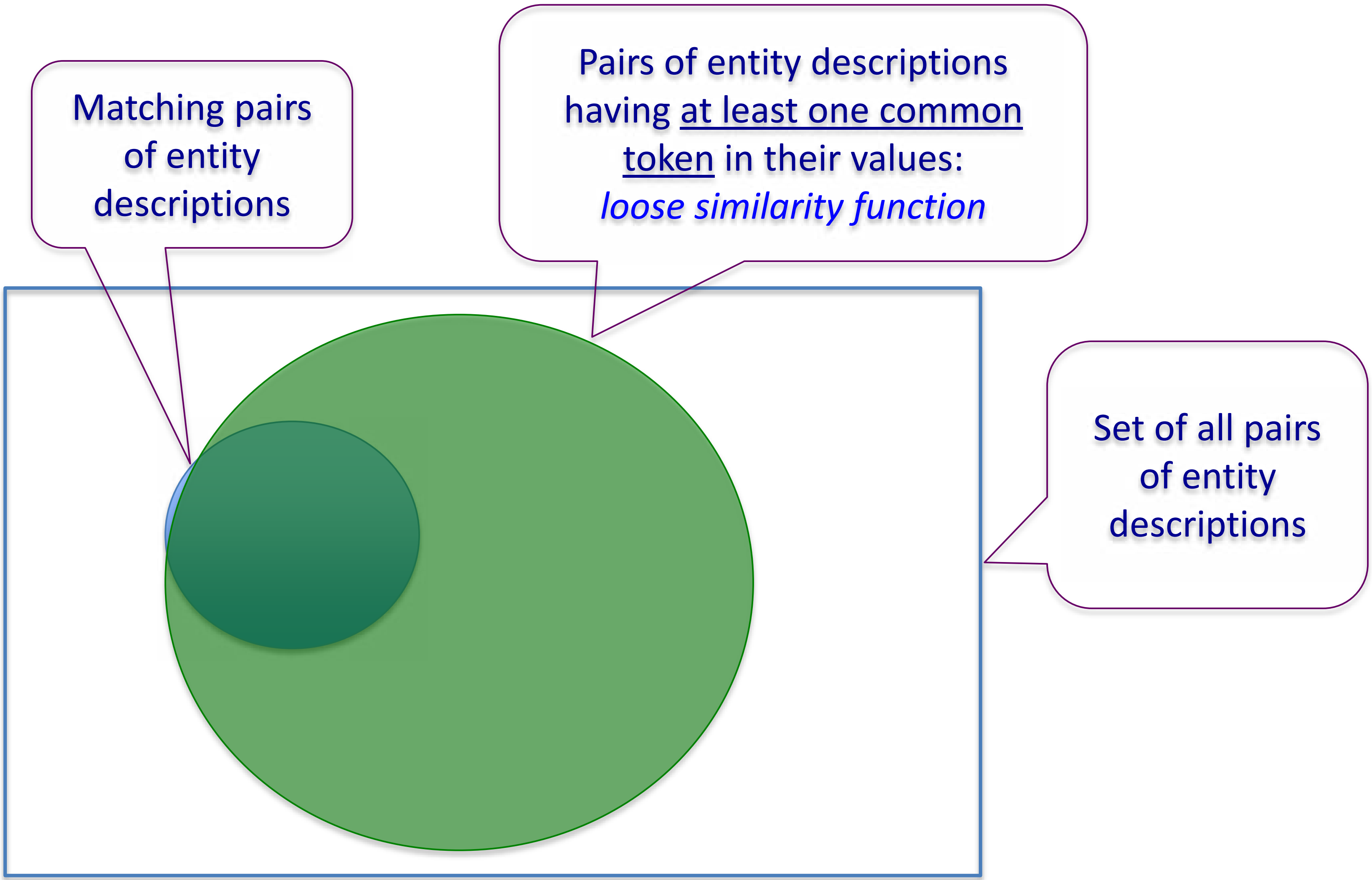
Token blocking achieves:

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

Token Blocking - Evaluation

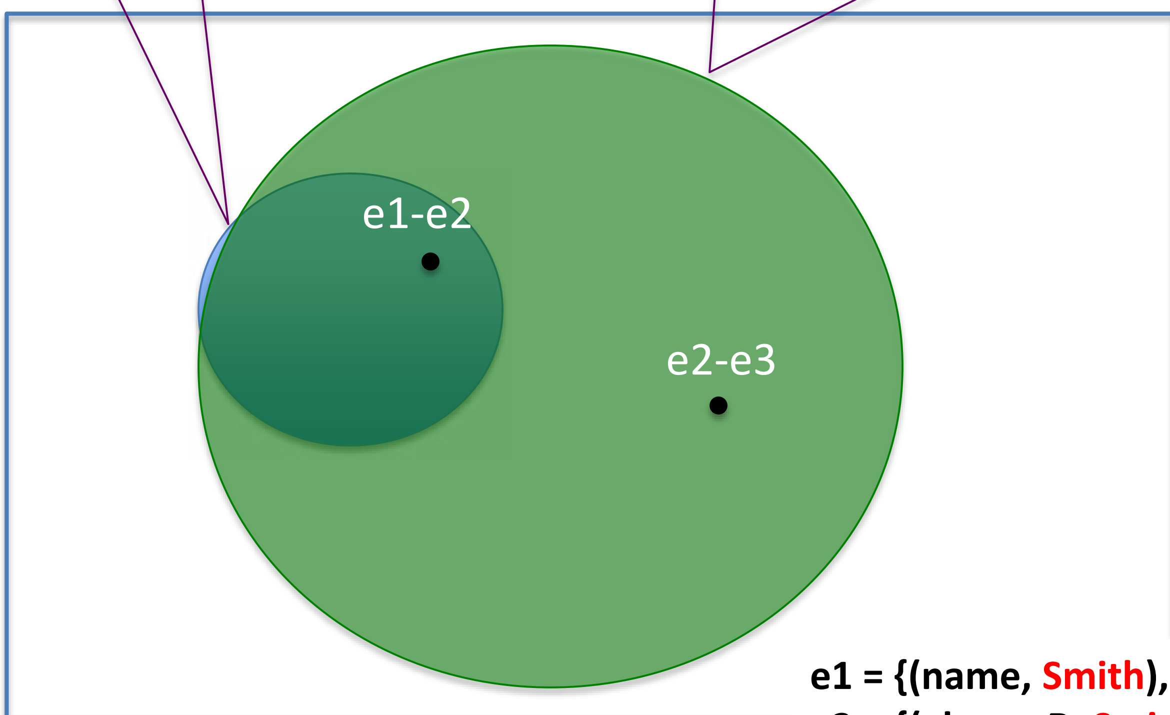


A single common token in the set of values is enough to place two descriptions in the same block

Token Blocking - Evaluation

Matching pairs of entity descriptions

Pairs of entity descriptions having <u>at least one common token</u> in their values: *loose similarity function*



Set of all pairs of entity descriptions

e1 = {(name, Smith), (country, USA)}

e2 = {(about, R. Smith), (livesIn, California)}

e3 = {(title, California Dreamin'), (length, 2:34)}

Is this enough?

Token blocking totally ignores the valuable information of attribute names

To improves this, attribute clustering considers patterns in the values

[Papadakis et al. 2013 (a)]

Attribute Clustering Blocking [Papadakis et al. 2013 (a)]

The goal again is to identify matches between two datasets, D_1 and D_2, each containing no duplicates – Clean-Clean Entity Resolution

Two main steps:

1. Similar attributes are placed together in non-overlapping clusters
2. Token blocking is performed on the descriptions of each cluster

Creating Clusters of Attributes

1. For each attribute of dataset D_1:
   - Find the most similar attribute of dataset D_2
2. For each attribute of dataset D_2:
   - Find the most similar attribute of dataset D_1
3. Compute the transitive closure of the generated pairs of attributes
4. Connected attributes form clusters
5. All single-member clusters are merged into a common cluster

Similarities between attributes are computed wrt. the string similarities of the values appearing in these attributes

Creating Clusters of Attributes

about	Eiffel Tower
architect	Sauvestre
year	1889
located	Paris

e11

about	Statue of Liberty
architect	Bartholdi Eiffel
year	1886
located	NY

e12

about	Auguste Bartholdi
born	1834

e13

about	Joan Tower
born	1938

e14

work	Lady Liberty
artist	Bartholdi
location	NY

e15

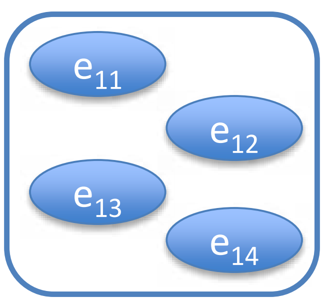
work	Eiffel Tower
year-constructed	1889
location	Paris

e16

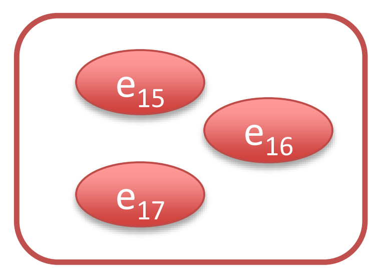
work	Bartholdi Fountain
year-constructed	1876
location	Washington D.C.

e17

D1



D2



Clustering Attributes: Example

about	Eiffel Tower	
architect	Sauvestre	
year	1889	
located	Paris	e11

about	Statue of Liberty	
architect	Bartholdi Eiffel	
year	1886	
located	NY	e12

about	Auguste Bartholdi	
born	1834	e13

about	Joan Tower	
born	1938	e14

work	Lady Liberty	
artist	Bartholdi	
location	NY	e15

work	Eiffel Tower	
year-constructed	1889	
location	Paris	e16

work	Bartholdi Fountain	
year-constructed	1876	e17
location	Washington D.C.	

Finding the attribute of D2 that is the most similar to the attribute "about" of D1:

values of about: {Eiffel, Tower, Statue, Liberty, Auguste, Bartholdi, Joan}

compared to (with Jaccard similarity) :

values of <u>**work**</u>: {Lady, Liberty, Eiffel, Tower, Bartholdi, Fountain} → **Jaccard = 4/9**

values of artist: {Bartholdi} → Jaccard = 1/8

values of location: {NY, Paris, Washington, D.C.} → Jaccard = 0

values of year-constructed: {1889, 1876} → Jaccard = 0

Clustering Attributes: Example

about	Eiffel Tower
architect	Sauvestre
year	1889
located	Paris

e11

about	Statue of Liberty
architect	Bartholdi Eiffel
year	1886
located	NY

e12

about	Auguste Bartholdi
born	1834

e13

about	Joan Tower
born	1938

e14

work	Lady Liberty
artist	Bartholdi
location	NY

e15

work	Eiffel Tower
year-constructed	1889
location	Paris

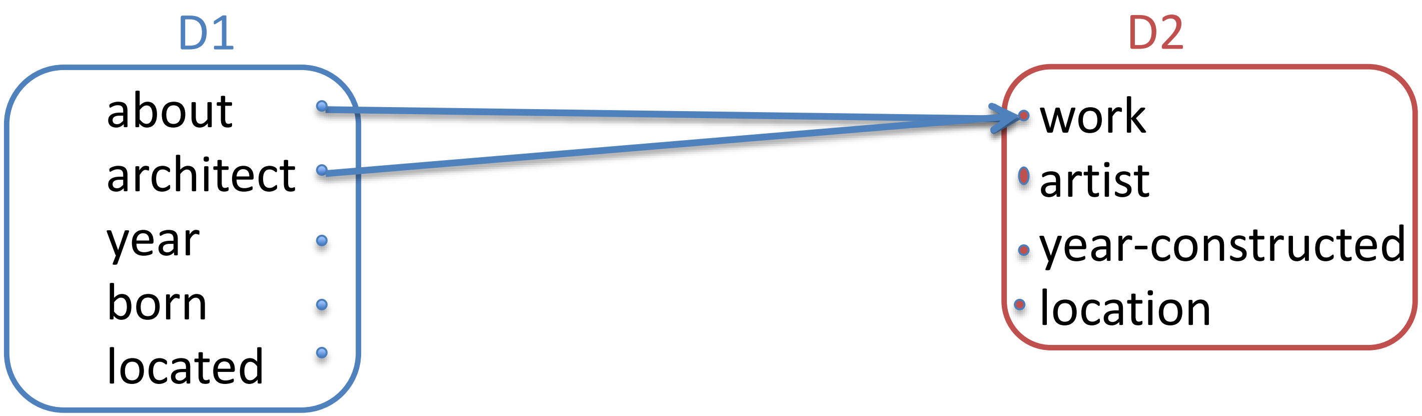
e16

work	Bartholdi Fountain
year-constructed	1876
location	Washington D.C.

e17

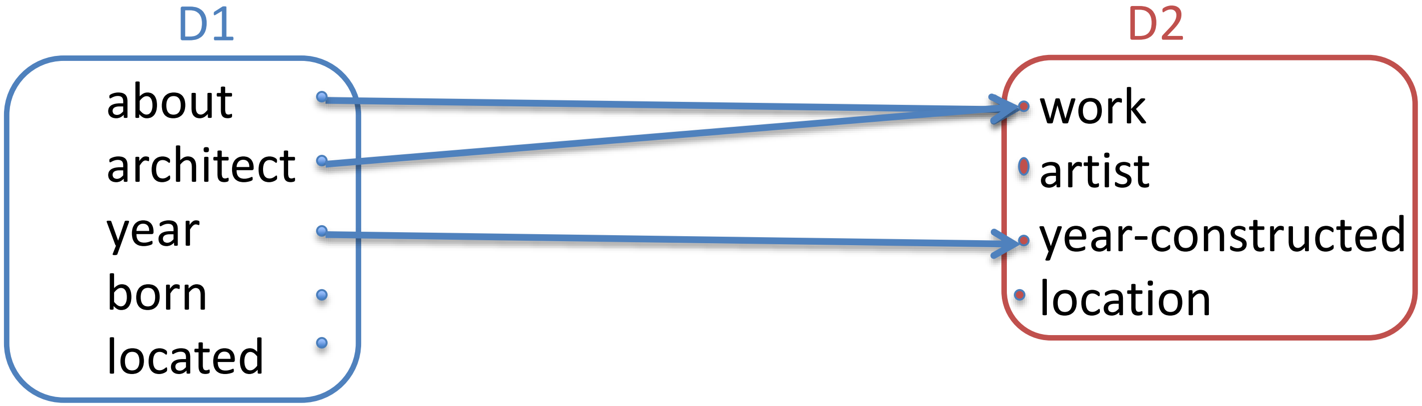
Clustering Attributes: Example

Similarly for the rest of the attributes…



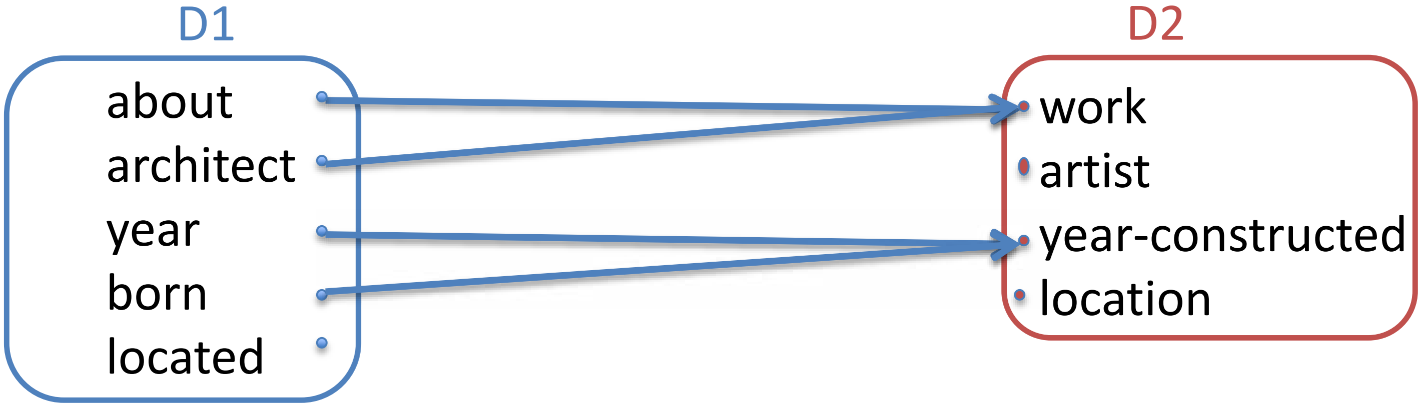
Clustering Attributes: Example

Similarly for the rest of the attributes…



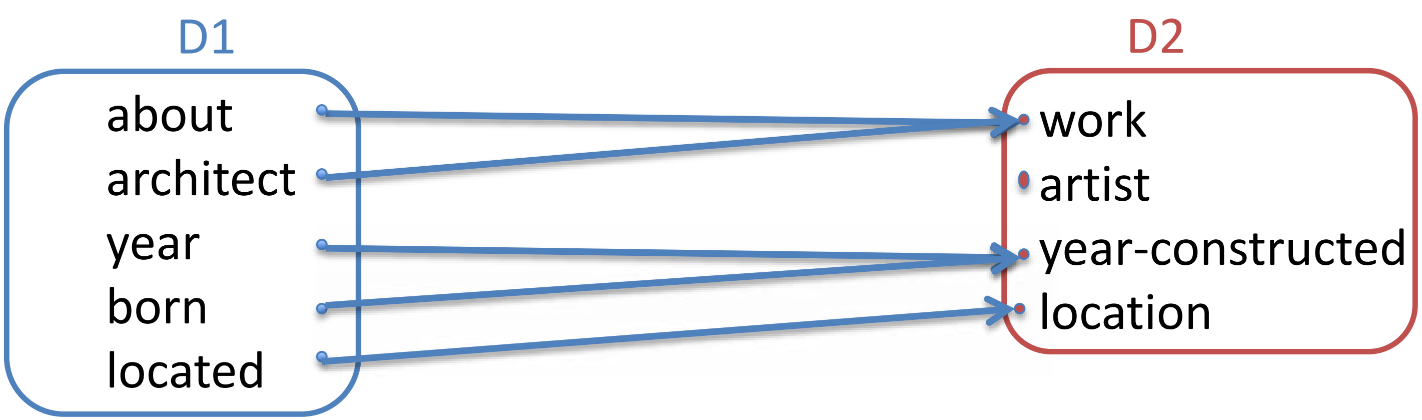
Clustering Attributes: Example

Similarly for the rest of the attributes…



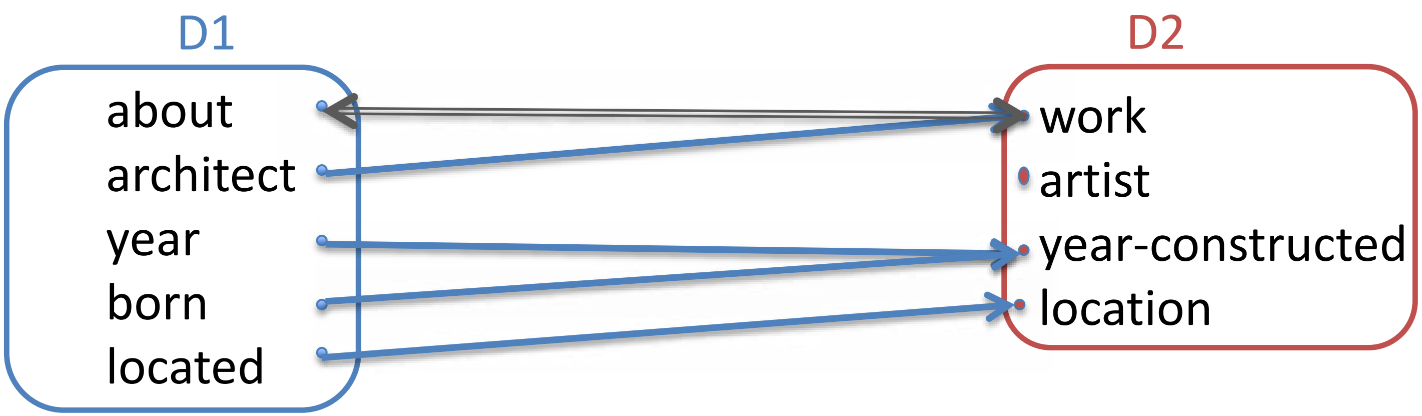
Clustering Attributes: Example

Similarly for the rest of the attributes…



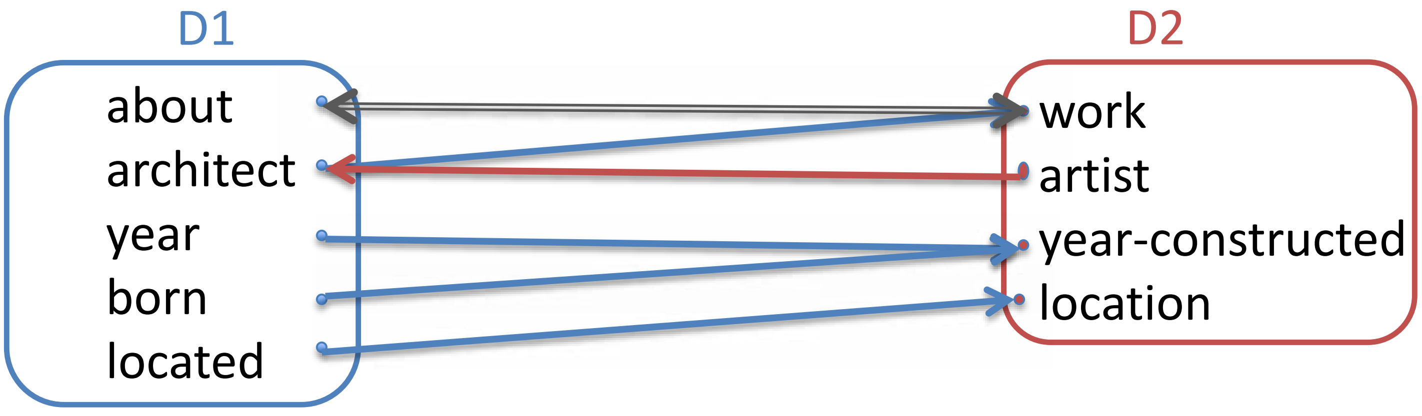
Clustering Attributes: Example

Similarly for the rest of the attributes…



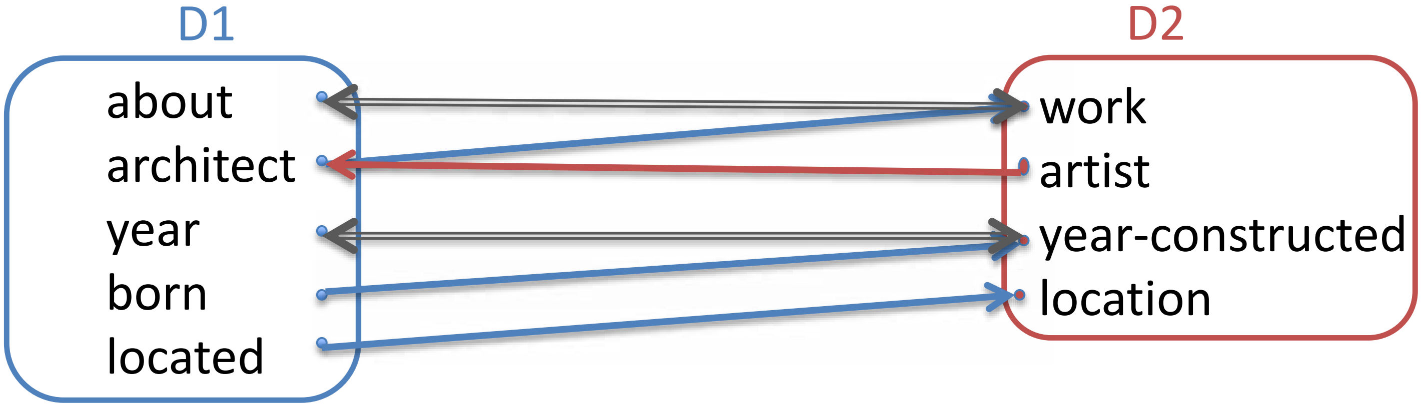
Clustering Attributes: Example

Similarly for the rest of the attributes…



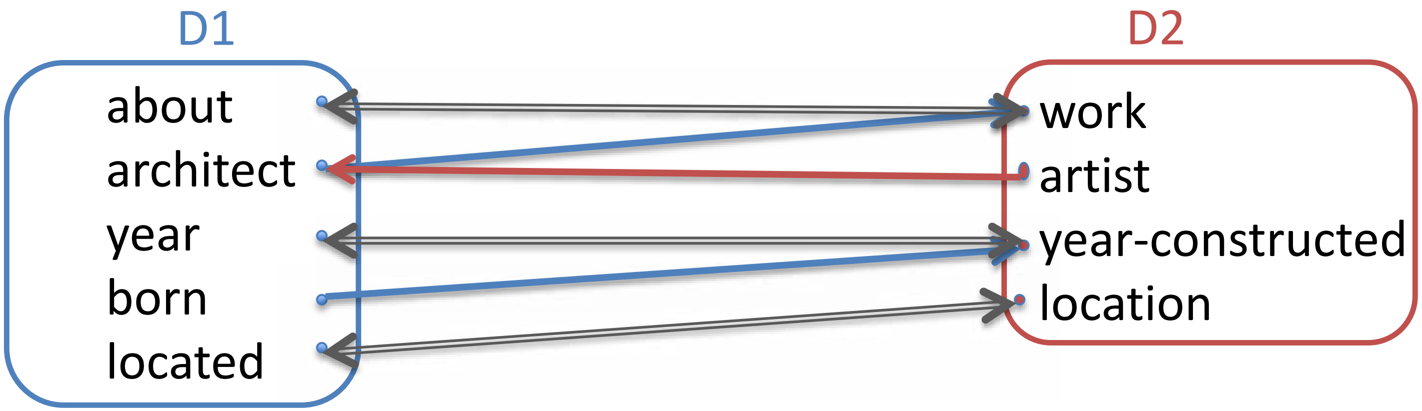
Clustering Attributes: Example

Similarly for the rest of the attributes…



Clustering Attributes: Example

Similarly for the rest of the attributes…



Clustering Attributes: Example

about	Eiffel Tower
architect	Sauvestre
year	1889
located	Paris

e11

about	Statue of Liberty
architect	Bartholdi Eiffel
year	1886
located	NY

e12

about	Auguste Bartholdi
born	1834

e13

about	Joan Tower
born	1938

e14

work	Lady Liberty
artist	Bartholdi
location	NY

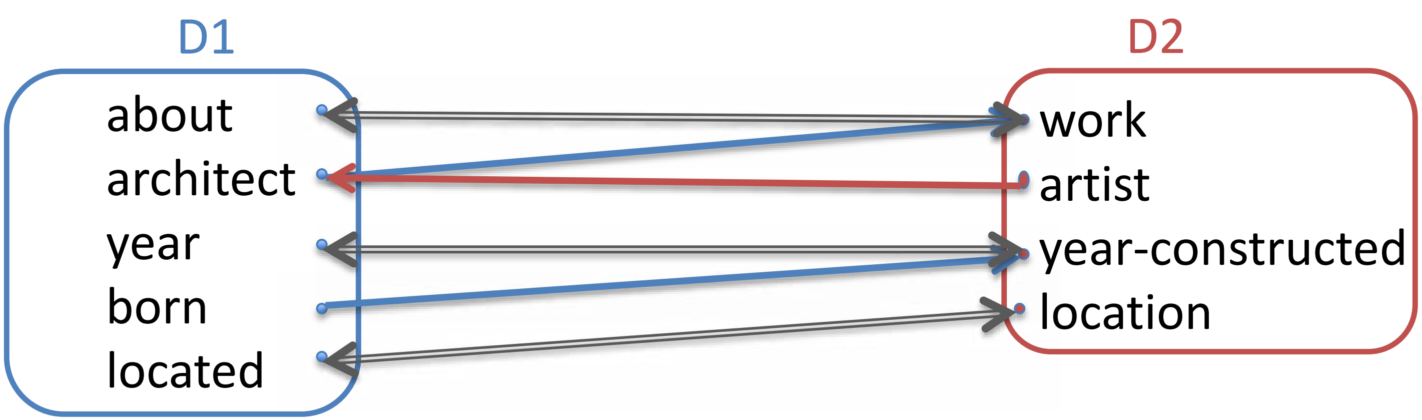
e15

work	Eiffel Tower
year-constructed	1889
location	Paris

e16

work	Bartholdi Fountain
year-constructed	1876
location	Washington D.C.

e17



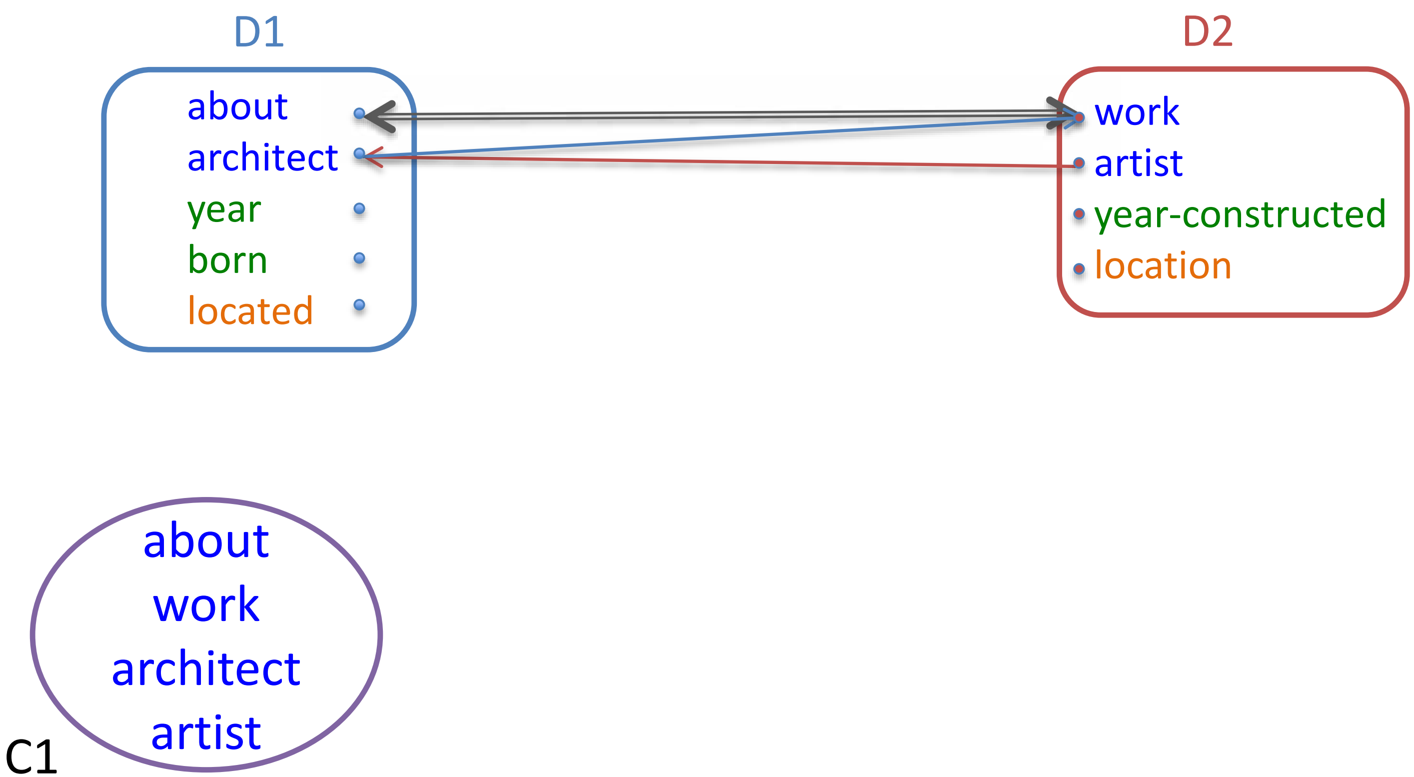
Clustering Attributes: Example

Compute the transitive closure of the generated attribute pairs

- Connected attributes form clusters

Pairs: (about, work), (work, about), (artist, architect), (architect, work)

Transitive closure:



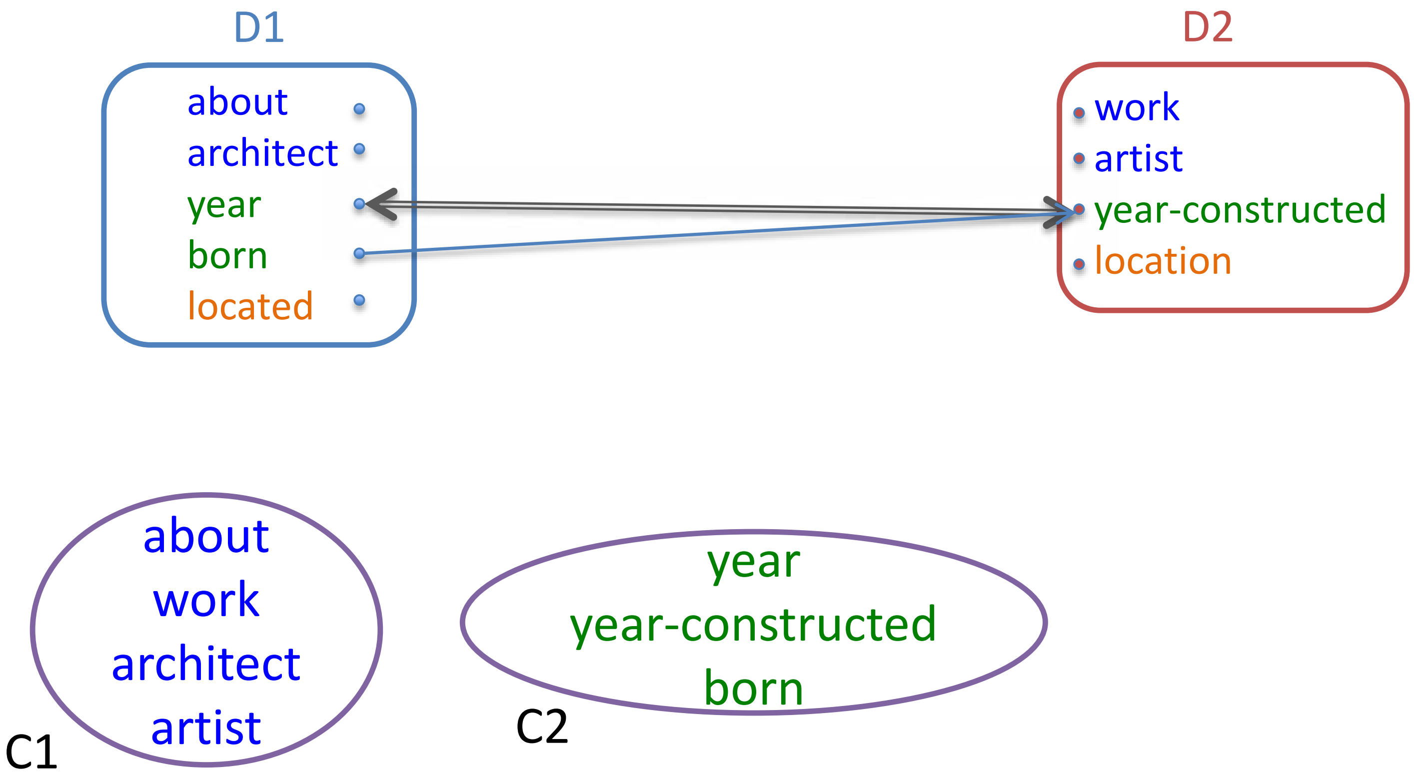
Clustering Attributes: Example

Compute the transitive closure of the generated attribute pairs

- Connected attributes form clusters

Pairs: (year, year-constructed), (year-constructed, year), (year-constructed, born)

Transitive closure:



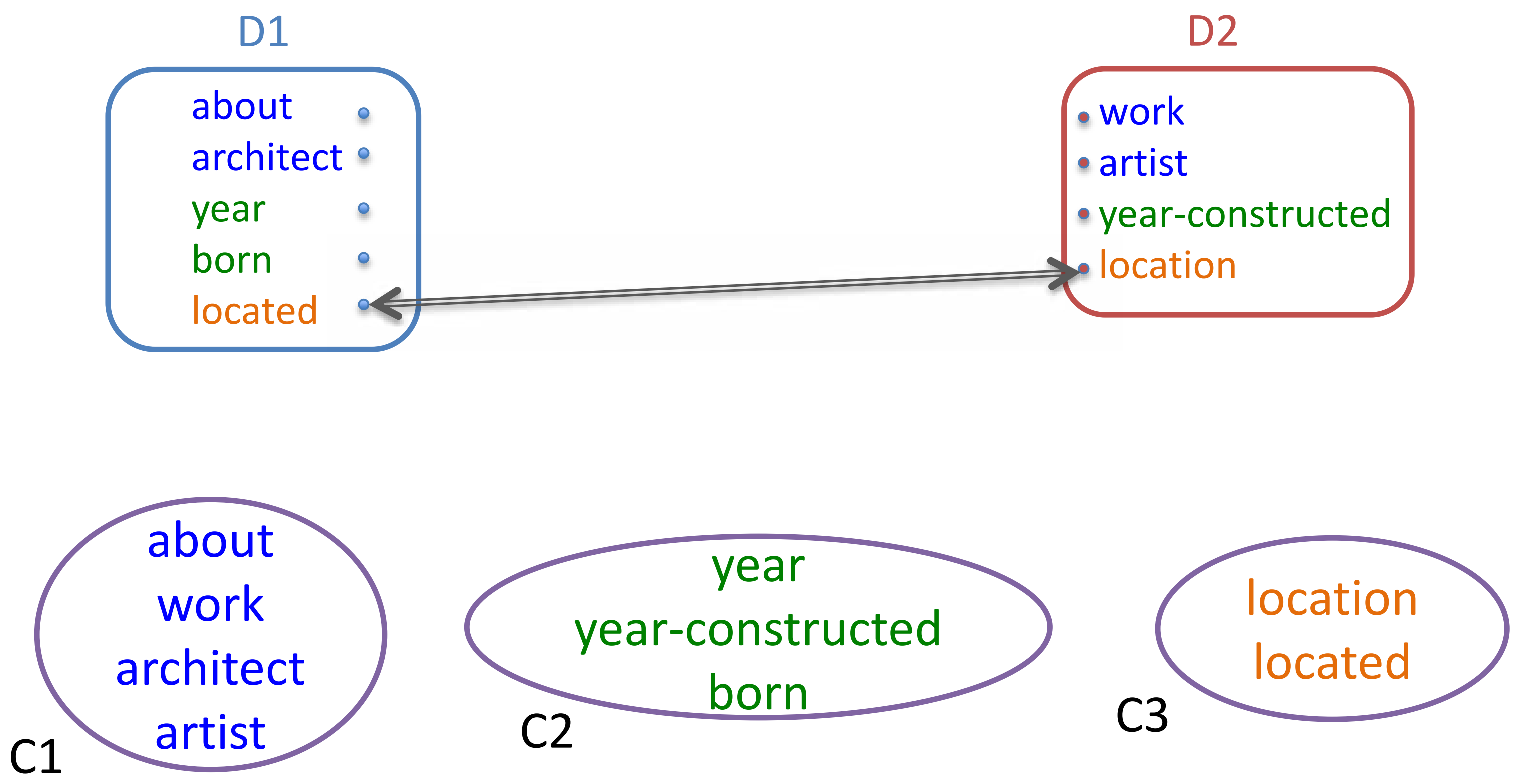
Clustering Attributes: Example

Compute the <u>transitive closure</u> of the generated attribute pairs

– Connected attributes form clusters

Pairs: (located, location), (location, located)

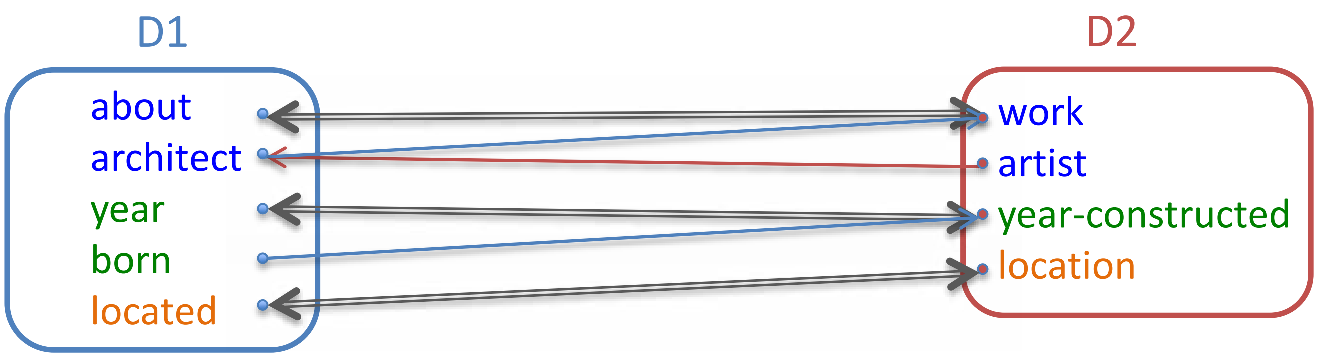
Transitive closure:



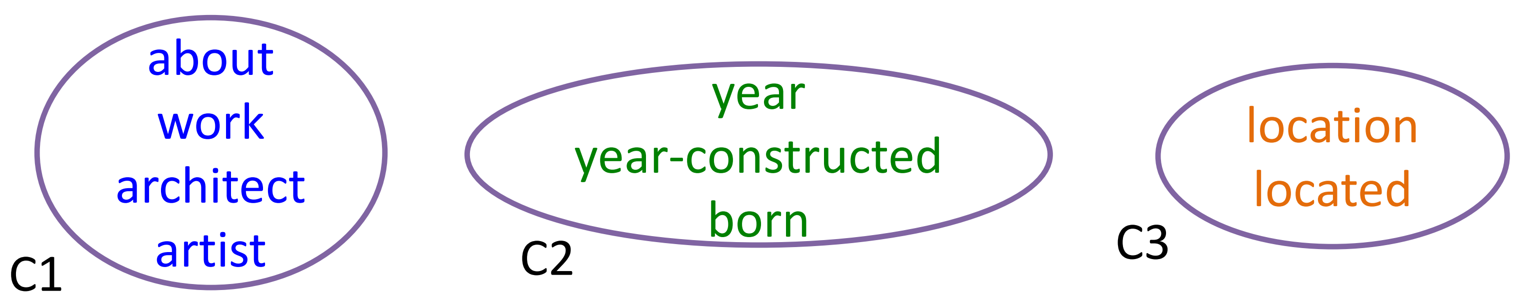
Clustering Attributes: Example

Compute the transitive closure of the generated attribute pairs

- Connected attributes form clusters



Generated attribute clusters:



Token Blocking for Each Cluster

e11

about	Eiffel Tower
architect	Sauvestre
year	1889
located	Paris

e12

about	Statue of Liberty
architect	Bartholdi Eiffel
year	1886
located	NY

e13

about	Auguste Bartholdi
born	1834

e14

about	Joan Tower
born	1938

e15

work	Lady Liberty
artist	Bartholdi
location	NY

e16

work	Eiffel Tower
year-constructed	1889
location	Paris

e17

work	Bartholdi Fountain
year-constructed	1876
location	Washington D.C.

C1
- about
- work
- architect
- artist

C2
- year
- year-constructed
- born

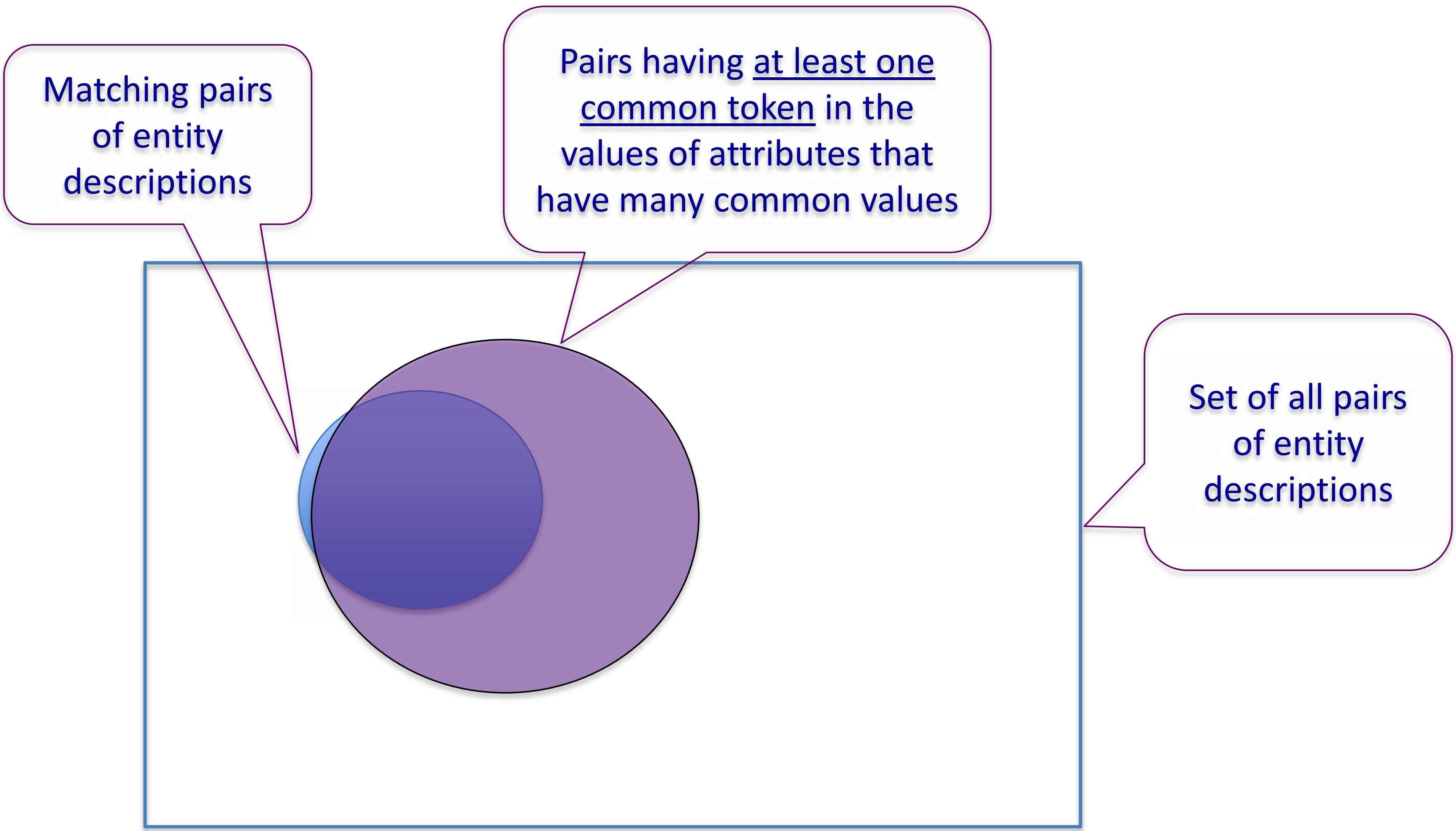
C3
- location
- located

Some of the generated blocks:

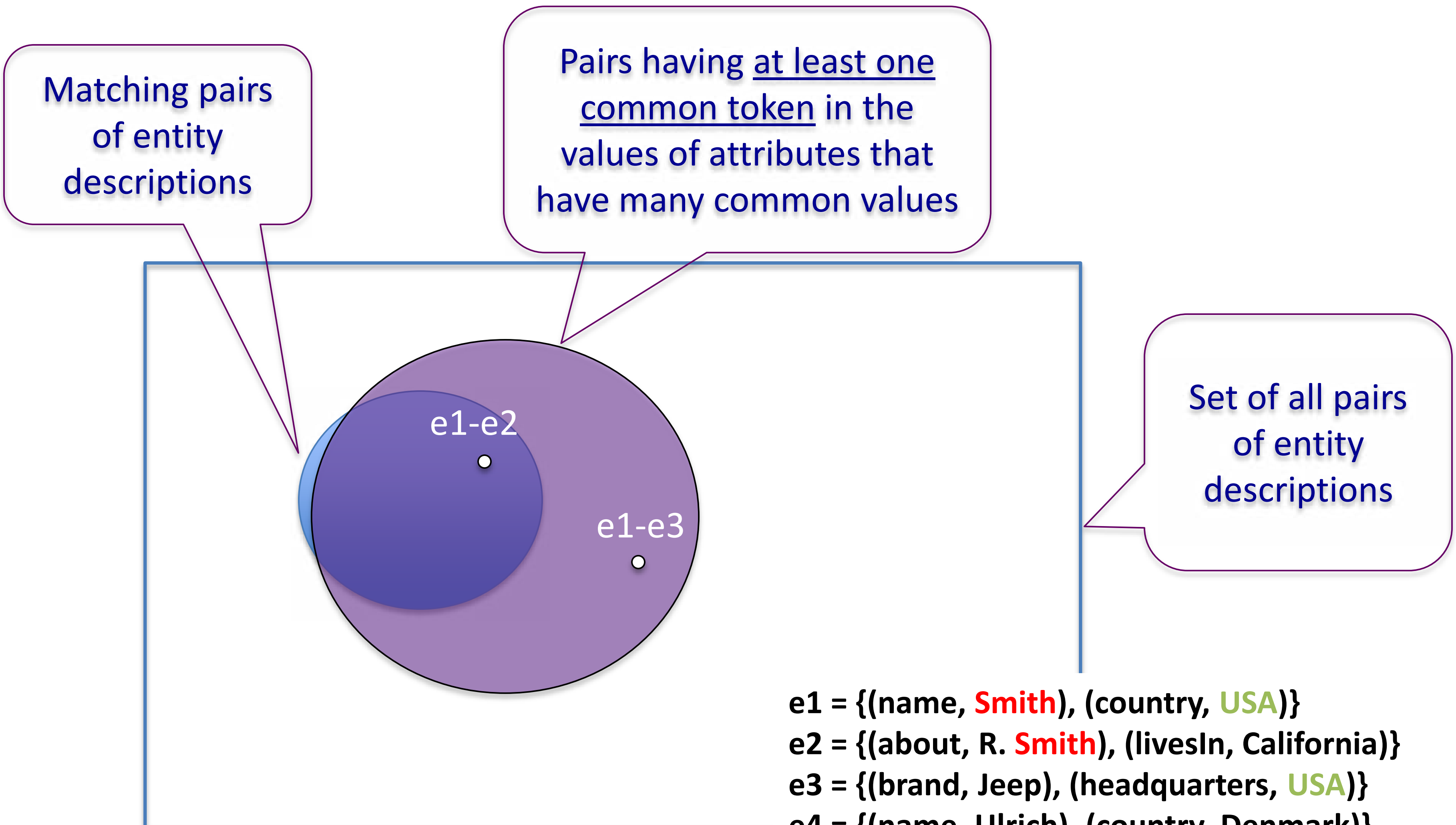
C3.NY	C1.Tower	C1.Bartholdi
e_{12}, e_{15}	e_{11}, e_{14}, e_{16}	e_{12}, e_{13}, e_{15}, e_{17}

→ *compare Lady Liberty to Auguste Bartholdi*

Attribute Clustering Blocking- Evaluation



Attribute Clustering Blocking- Evaluation



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)}

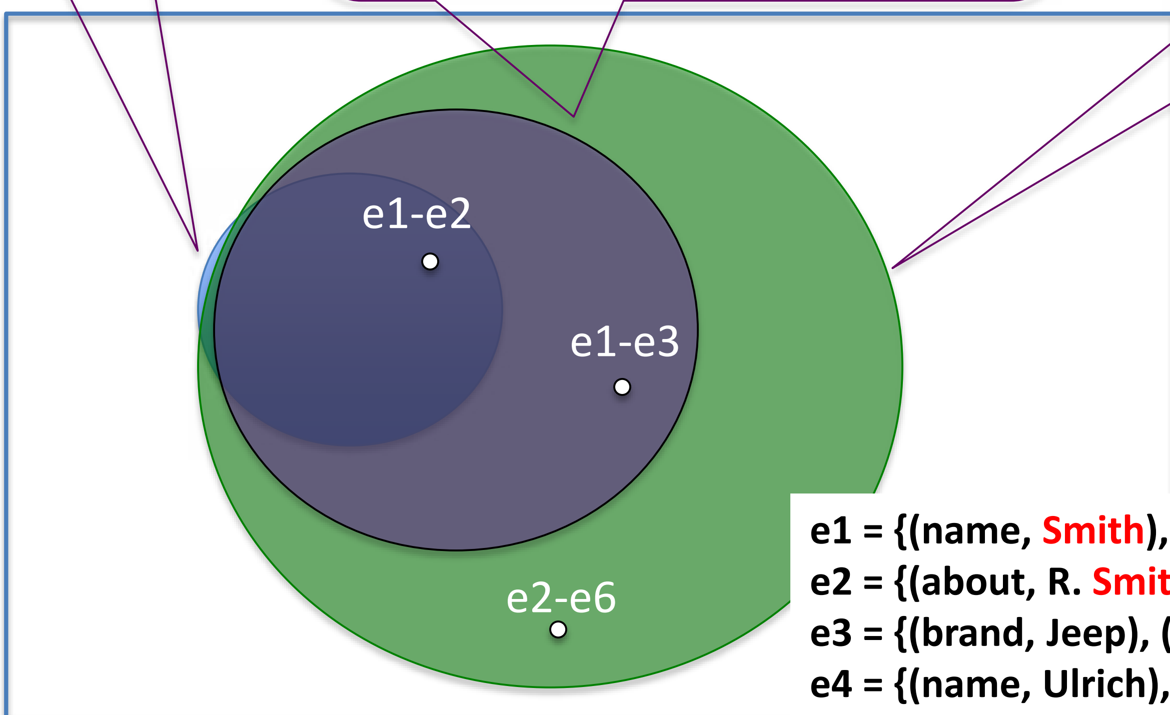
Attribute Clustering Blocking vs Token Blocking

Matching pairs of entity descriptions

Pairs having <u>at least one common token</u> in the values of attributes that have many common values: *a not so loose similarity function*

Pairs having <u>at least one common token</u> in their values: *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)}

Attribute Clustering Blocking vs Token Blocking

In attribute clustering:

- High recall
- Better efficiency compared to token blocking (save many redundant comparisons)
- Low precision

Many non-matches are placed in the same block

The same pair of descriptions is contained in many blocks

Much more expensive to build the blocks, than just performing token blocking

Again, it ignores the valuable semantics that attributes and entity relationships offer

ZenCrowd [Demartini et al. 2013]

A different approach to attribute clustering

Three-stage blocking:

1. Token blocking on the labels of the descriptions

2. Rank description pairs within blocks, based on the Jaccard similarity of the values of matching attribute pairs
   - Attribute matching is based on the number of exact string matches that two attributes have in their values (within block)

3. Ask humans for the low-ranked pairs (crowdsourcing)

Find this Target Entity:
Spoleto (Italy)

- ○ Ariulf of Spoleto
- ○ Spoleto Festival, Italy
- ○ Spoleto
- ○ Spoleto Festival (taped in Italy): Sir John Gielgud; Eileen Farrell
- ○ Winiges of Spoleto

☐ No Result is the same as the Target Entity

ZenCrowd - Example

name	Statue of Liberty
architect	Bartholdi Eiffel
year	1886
located	NY

e1

about	Lady liberty
architect	Eiffel
location	NY

e2

about	Eiffel Tower
architect	Sauvestre
year	1889
location	Paris

e3

1. token blocking on the labels of the descriptions

Statue	Liberty	Lady	Eiffel	Tower
e_1	e_1, e_2	e_2	e_3	e_3

=> Pairs: $\{(e_1, e_2)\}$

2. attribute matching (only between e_1 and e_2):
   - #exact string matches(name, about) = 1 ("Liberty")
   - #exact string matches(architect, architect) = 1 ("Eiffel")
   - #exact string matches(architect, location) = 0
   - #exact string matches(year, architect) = 0
   - …
   - #exact string matches(located, location) = 1 ("NY")
     - matching attribute-pairs:
       (name, about), (architect, architect), (located, location)

J(name, about) =
J({Statue, Liberty}, {Lady, Liberty})
= 1/3

similarity(e_1, e_2)=(J(located, location) + J(architect, architect) + J(name, about)) /3
= (1 + 1/2 + 1/3) / 3 = 0.61

Blocking in the Web of Data

Technique	Put two descriptions in a common block, when they have…
Token Blocking	a common token in their values
Attribute Clustering Blocking	a common token in the values of attributes that have similar values in overall
ZenCrowd	on average, similar values for attributes that have similar values in overall

An entity resolution task can also receive only one (Dirty) entity collection as input

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	m.072p8	5139572	Tina_Brown
e_1, e_2	e_3	e_4	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:

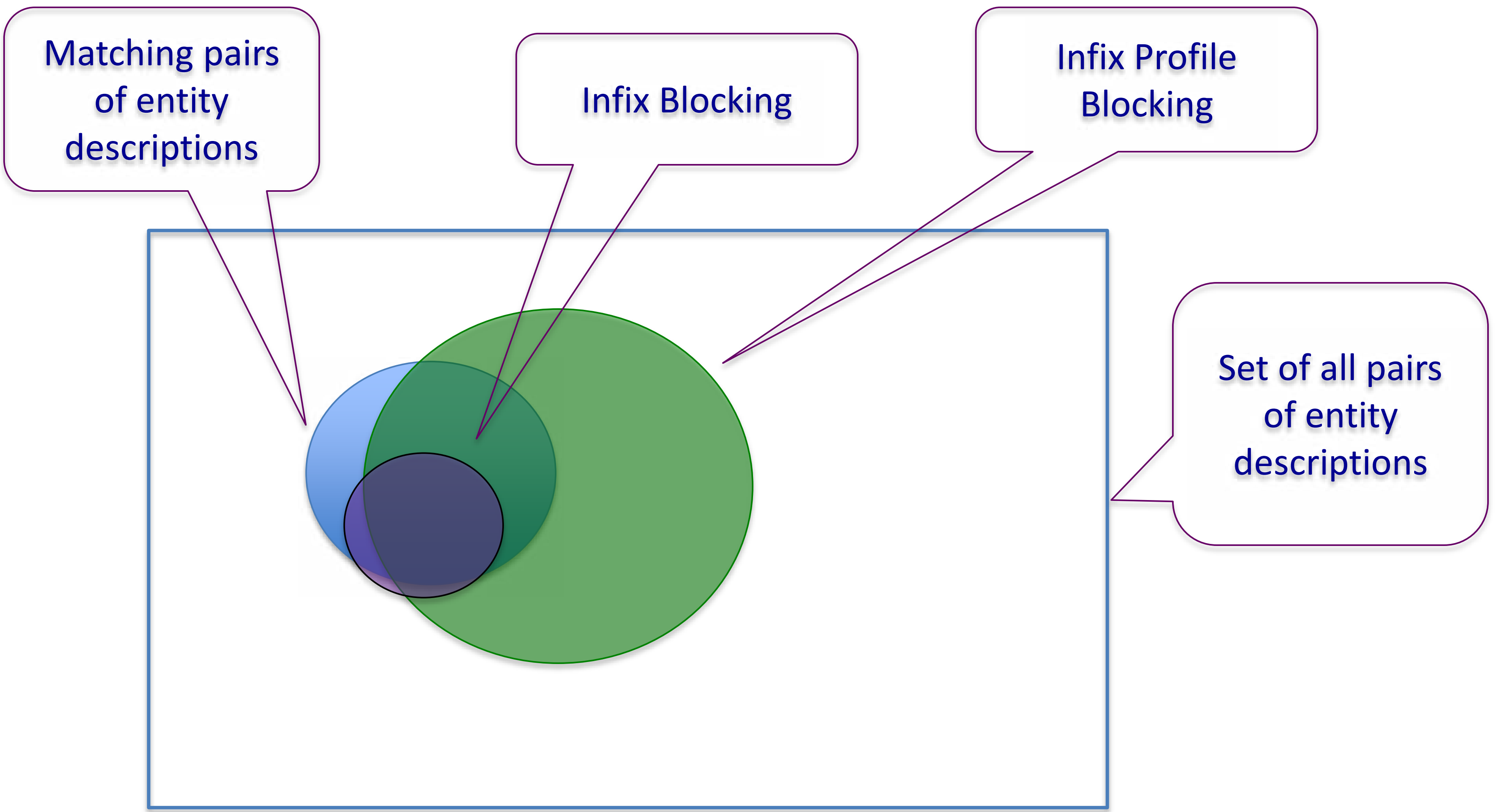
Liberty_Island
e_1, e_2, e_3, e_5

m.026kp2
e_3

5124330
e_4

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

Prefix-Infix(-Suffix) - Evaluation



Blocking in the Web of Data

Technique	Put two descriptions in a common block, when they have…
Token Blocking	a common token in their values
Attribute Clustering Blocking	a common token in the values of attributes that have similar values in overall
ZenCrowd	on average, similar values for attributes that have similar values in overall
Prefix-Infix(-Suffix)	a common token in their literal values, or a common URI

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

<u>The goal</u>: *Reduce further the number of comparison*

Block Post-Processing

- 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
    - Wrt. the number of superfluous comparisons spared in subsequently examined blocks
- 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
    - Based on the number of blocks they appear together [Papadakis et al. 2011b]
- Remove low-order comparisons [Whang et al. 2013, Papadakis et al. 2011b]
  - Similar to removing low-order blocks

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

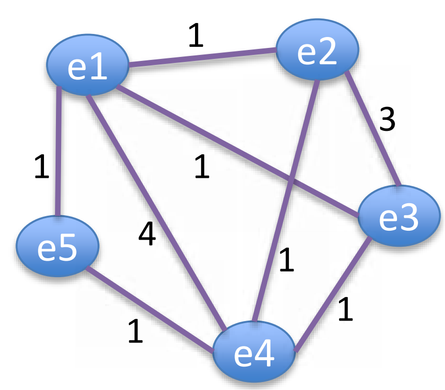
Blocks:
(with token blocking)

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

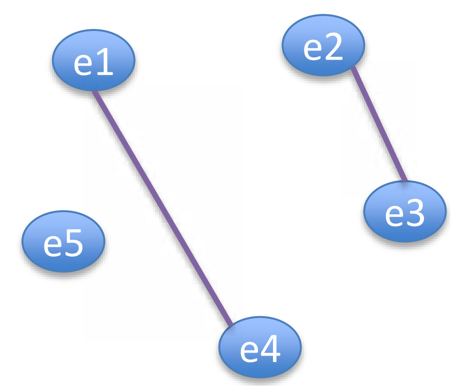
13 comparisons
to identify 2 matches

Blocking graph:



edge weights = #common blocks

Pruned blocking graph:
(remove edges with weight < 2)



2 comparisons
to identify 2 matches

Conclusions of Part I

Partitioning vs. Overlapping Blocks

Blocking approaches can be distinguished between:

- Partitioning: Each description is placed in exactly one block
  - Fewer comparisons
- Overlapping: Each description is placed in more than one block
  - More identified matches

Selecting a good blocking key is more important than the blocking technique

[Christen 2012]

In the Web of Data, selecting a (good) blocking key is not straightforward!

Discussion on Blocking

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

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

Tutorial Overview

- Iterative entity resolution approaches
  - Coffee break!

What follows in Part II:

- Continue on iterative entity resolution approaches
- Large scale entity resolution using MapReduce
- Conclusions

Iterative Approaches

Iterative Entity Resolution

Basic algorithm for entity resolution in one source E (dirty)

- Compare each entity description $e_i \in S$ with all other entity descriptions in E, i.e., with all $e_j \in E \setminus \{e_i\}$
- For comparison, use a match function to classify each pair (e_i, e_j) as a match/non-match
  - Based on similarity measures
  - Based on domain-specific rules
  - Based on a combination of both
- Complexity: $O(N^2)$, with N being the number of entity descriptions in E

Algorithm easily extends to entity resolution among two sources (clean-clean or dirty-dirty)

Iterative Entity Resolution

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

Tutorial Overview

What follows in Part II:

- Continue on iterative entity resolution approaches
- Large scale entity resolution using MapReduce
- Conclusions