dbTrento

Entity Resolution for BIG Data

Blocking-based Entity Resolution in Highly Heterogeneous Information Spaces

George Papadakis
L3S Research Center

Themis Palpanas
University of Trento

Acknowledgements

Ekaterini Ioannou

Claudia Niederee

Wolfgang Nejdl

Peter Fankhauser

Data Evolution

structured

precise

static (almost)

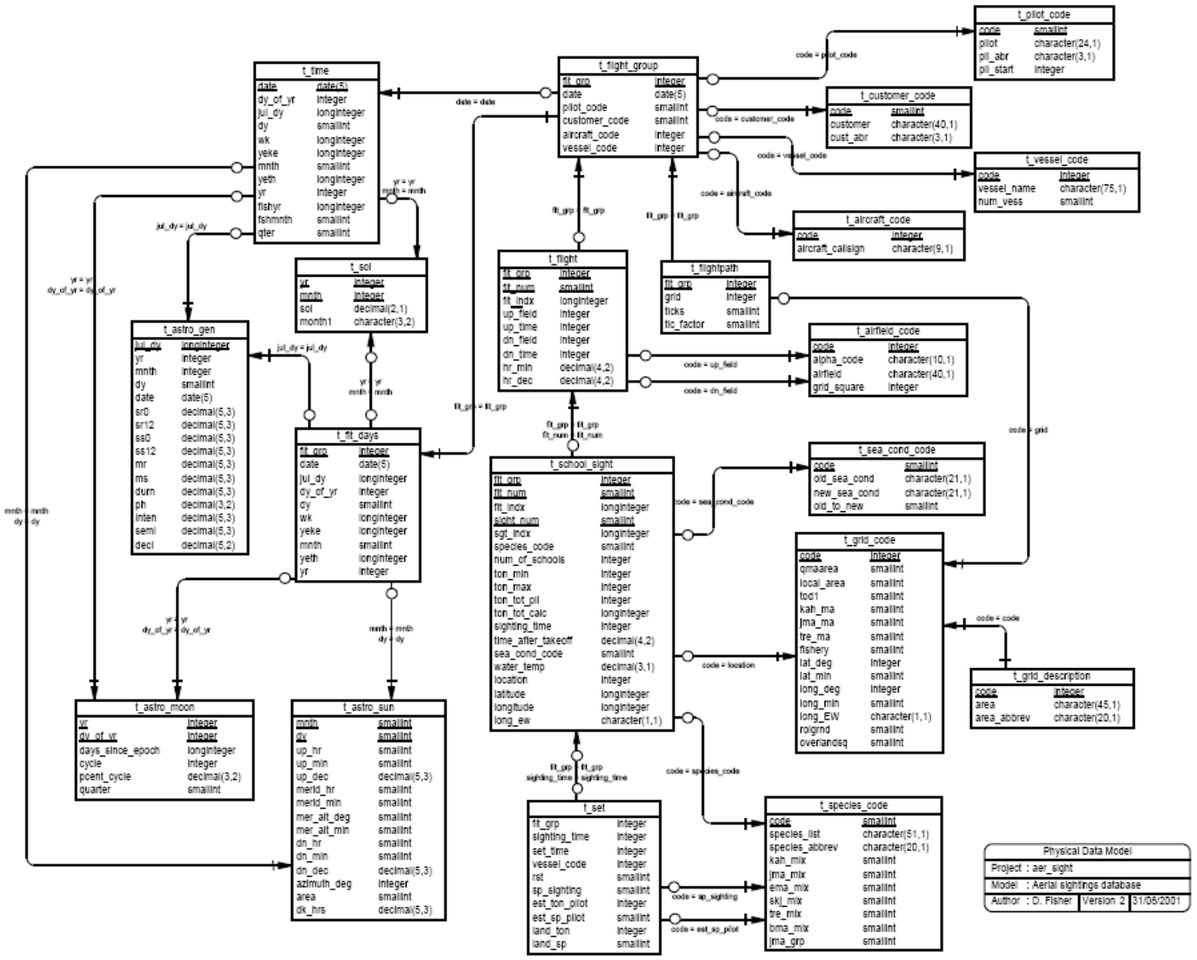


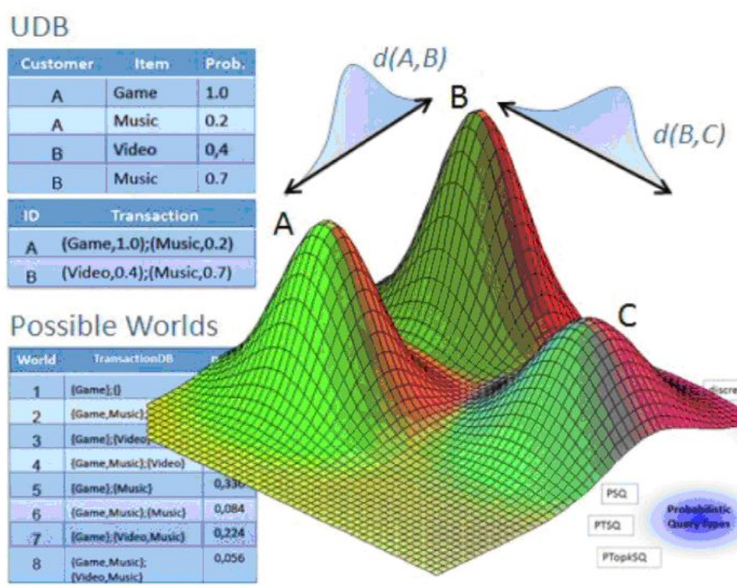
Figure 1: Entity Relationship Diagram (ERD) for the aer_sight database

Data Evolution

non-structured

uncertain

streaming



Things to Come

large scale data

- very large collections (i.e., terabytes to exabytes)
- scientific, business, user generated

Things to Come

large scale data

streaming data

- sensors, feeds, continuous analytics
- response times in seconds to nanoseconds

Things to Come

large scale data

streaming data

heterogeneous data

- structured, non-structured, text, multimedia
- variety of sources, schemas, representations, models
- computer-generated, human-generated

Things to Come

large scale data

streaming data

heterogeneous data

private data

- correlating data from multiple sources poses risks
  - credit card history, mobile phone usage, GPS tracking
- privacy and accountability
  - access granted only to specific person, at specific time, for specific purpose, only for necessary data

Things to Come

large scale data

streaming data

heterogeneous data

private data

uncertain data

- imprecision, inconsistencies, incompleteness, ambiguities, latency, deception, approximations, privacy preserving transformations
- process/data/model uncertainty

Big Data

large scale data

streaming data

heterogeneous data

private data

uncertain data

confluence of all the above!

Work in dbTrento

large scale data

streaming data

heterogeneous data

private data

uncertain data

This Talk

large scale data

streaming data

heterogeneous data

private data

uncertain data

This Talk

large scale data (web data sources)

streaming data

heterogeneous data (no hope for standard schema)

private data

uncertain data (imprecisions, errors, typos)

Outline

Introduction

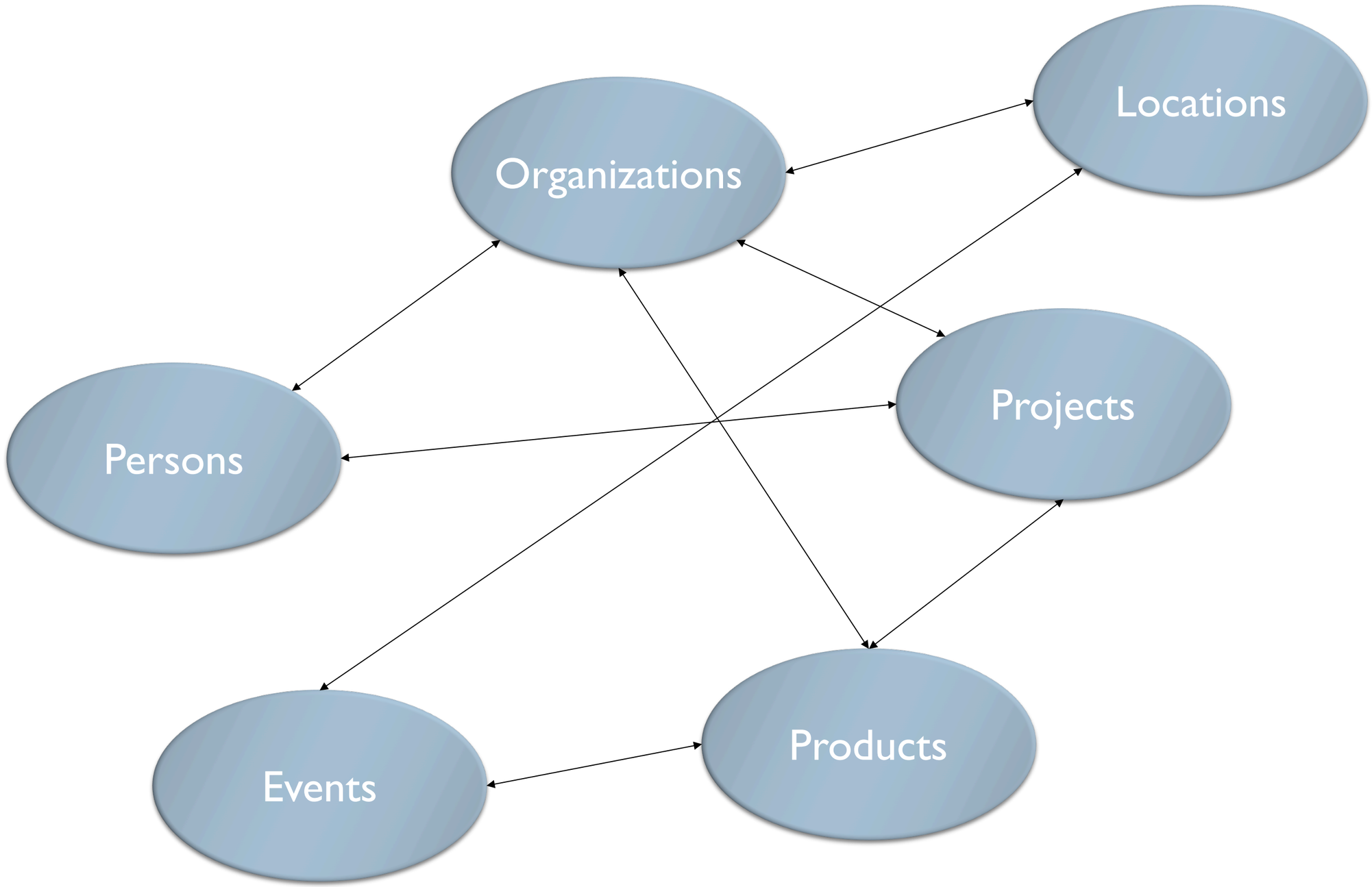
- Background
- Standard Blocking for Databases
- Advanced Blocking Methods
  - Q-grams Blocking
  - Suffix Array Blocking
  - The Sorted Neighborhood Approach
  - Canopy Clustering
- Blocking over Dataspaces

Approach

Evaluation

Entities: an invaluable asset

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



However …

How many names, descriptions or IDs (URIs) are used for the same "entity"?

London 런던 ⲗⲟⲛⲇⲟⲛ لندن لندن ლონდონი ለንደን ロンドン ლონდონ ลอนดอน இலண்டன் ლოնდოնი Llundain Londain Londe Londen Londen Londen Londinium London Londona Londonas Londoni Londono Londra Londres Londrez Londyn Lontoo Loundres Luân Đôn Lunden Lundúnir Lunnainn Lunnon نوندون لندن لندن لندن
Ο Λονδίνο Лёндан Лондан Лондон Лондон Лондон Лондон Լոնդոն 伦敦 …

capital of UK, host city of the IV Olympic Games, host city of the XIV Olympic Games, future host of the XXX Olympic Games, city of the Westminster Abbey, city of the London Eye, the city described by Charles Dickens in his novels, …

http://sws.geonames.org/2643743/
http://en.wikipedia.org/wiki/London
http://dbpedia.org/resource/Category:London
…

dbTrento

… or …

How many "entities" have the same name?

- London, KY
- London, Laurel, KY
- London, OH
- London, Madison, OH
- London, AR
- London, Pope, AR
- London, TX
- London, Kimble, TX
- London, MO
- London, MO
- London, London, MI
- London, London, Monroe, MI
- London, Uninc Conecuh County, AL
- London, Uninc Conecuh County, Conecuh, AL
- London, Uninc Shelby County, IN
- London, Uninc Shelby County, Shelby, IN
- London, Deerfield, WI
- London, Deerfield, Dane, WI
- London, Uninc Freeborn County, MN
- ...

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

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

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

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

- ...

… or …

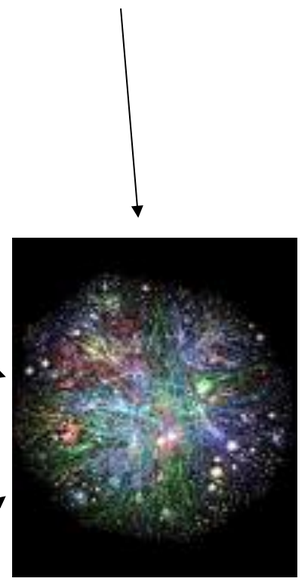
How many content types / applications provide valuable information about each of these "entities"?

News about London

REUTERS

reviews on hotels in London

Wiki pages about the London



Pictures and tags about London

flickr

LinkedIn

Social networks in London

YouTube
Broadcast Yourself

Videos and tags for London

Preliminaries

Entity Resolution:

identifies and aggregates the *different* entity profiles/records that actually describe the *same* real-world object.

Application areas:

Social Networks, census data, price comparison portals, Linked Data

Useful because:

- improves data quality and integrity
- fosters re-use of existing data sources

Types of input

The entity collections given as input to ER can be of two types:

- ***clean***, which are duplicate-free

  e.g., DBLP, ACM Digital Library, Wikipedia, Freebase,

- ***dirty***, which contain duplicate entity profiles in themselves

  e.g., Google Scholar, Citeseer^X

Types of Entity Resolution

An ER task that receives as input two entity collections can be of the following types:

1. ***Clean-Clean ER***

2. ***Dirty-Clean ER***

3. ***Dirty-Dirty ER***

Types of Entity Resolution

An ER task that receives as input two entity collections can be of the following types:

1. ***Clean-Clean ER*** (a.k.a. ***Record Linkage*** in databases)

   Given two clean, but overlapping entity collections, identify the entity profiles they have in common.

   e.g., merge DBLP with ACM Digital Library

2. ***Dirty-Clean ER***

3. ***Dirty-Dirty ER***

Types of Entity Resolution

An ER task that receives as input two entity collections can be of the following types:

1. ***Clean-Clean ER***
2. ***Dirty-Clean ER***
3. ***Dirty-Dirty ER***

Identify unique entity profiles contained in union of the input entity collections.

For simplicity, we treat them as equivalent to ***Dirty ER*** (a.k.a. ***Deduplication*** in databases), which receives a single dirty entity collection as input.

Computational cost

ER is an inherently quadratic problem (i.e., $O(N^2)$):
every entity has to be compared with all others

Does not scale to large entity collections (e.g., Big Data)

Solution: Blocking

- similar entities are grouped into blocks
- comparisons are only executed inside blocks

Assumptions for Blocking in Databases

Assumptions:

1. Each entity profile corresponds to a single real-world object.
2. Every entity profile consists of a uniquely identified set of name-value pairs.
3. A-priori known schema
4. For each attribute we know some metadata:
   - level of noise (e.g., spelling mistakes, false or missing values)
   - distinctiveness of values
5. Two matching profiles are detected if they have at least one block in common.

Standard Blocking for Databases

Simplest form of blocking.

Works as a hash function.

Algorithm:

1. Select the most appropriate attribute names w.r.t. noise and distinctiveness.
2. Extract a summary from their values in order to form a set of Blocking Key Values (BKVs)
3. For each BKV, create one block that contains all entities having this BKV in their transformation.

Example of Standard Blocking

DATASET 1

Entity 1
- first name=Antony P.
- last name=Gray
- address=Los Angeles, California
- zip_code= 91456

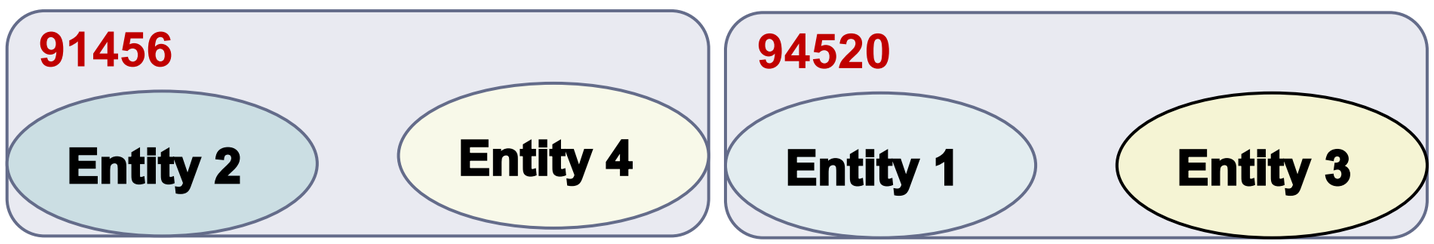
Entity 2
- first name=Bill
- last name=Green
- address=Los Angeles, California
- zip_code=94520

DATASET 2

Entity 3
- first name=Antony
- last name=Gray
- address=L.A., California, USA
- zip_code= 91456

Entity 4
- name=William Nicholas
- last name=Green
- address=L.A., California, USA
- zip_code=94520

Blocks on zip_code:



Advanced Blocking Methods

Most blocking methods for databases rely on the same signature-based functionality (review in [Christen, TKDE 2011]).

Main difference (to standard blocking):

redundancy → overlapping blocks

Drawbacks:

1. Schema-dependent
2. Too many parameters to be configured (e.g., which attributes to select, how to combine them in BKVs)

Partial solution:

automatic configuration through machine learning techniques

[Bilenko et. al., ICDM 2006] [Michelson et. al., AAAI 2006]

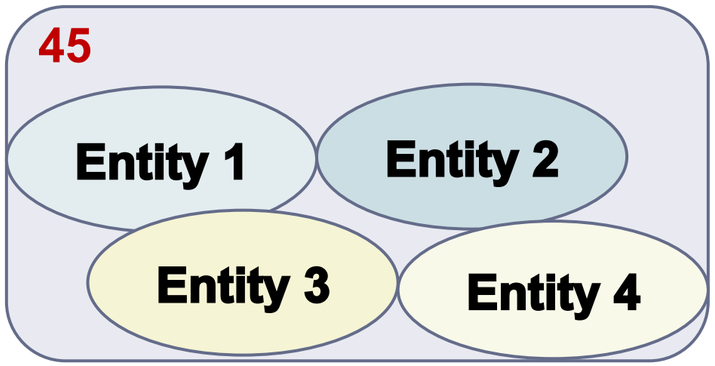
Q-grams Blocking

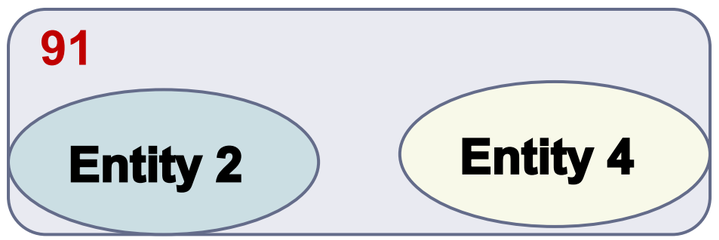
[Baxter et. al., KDD 2003] [Gravano et. al., VLDB 2001]

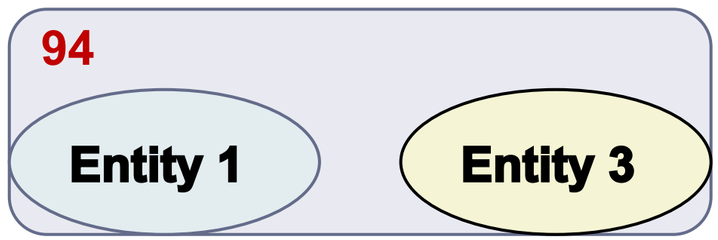
Converts every BKV into a list of *q*-grams

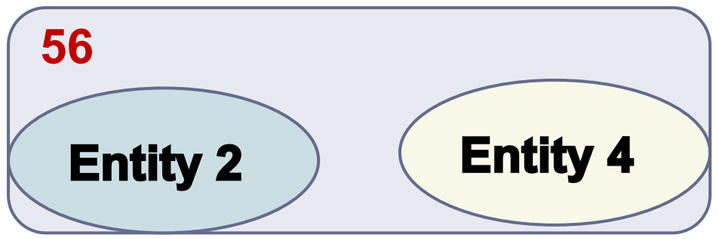
(i.e., substrings of length *q*).

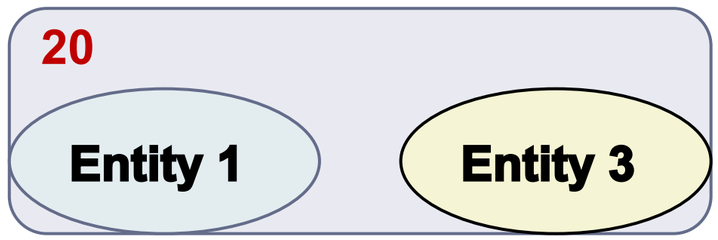
For *q*=2, the keys *91456* and *94520* yield the following blocks:

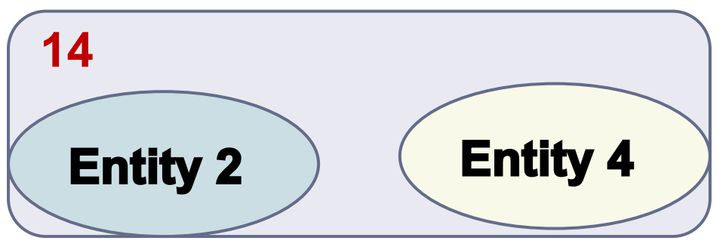


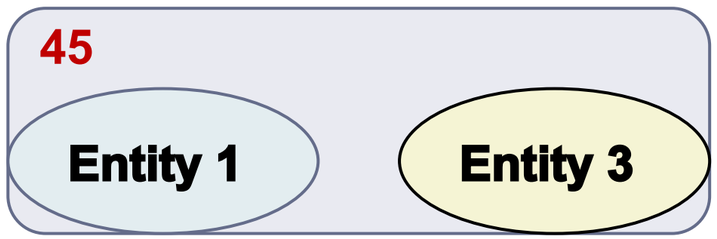












Advantage:

robust to noisy BKVs

Drawback:

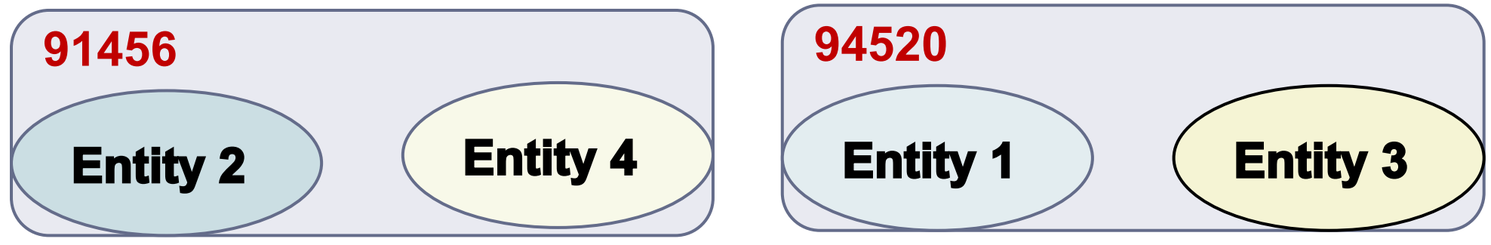
larger blocks → higher computational cost

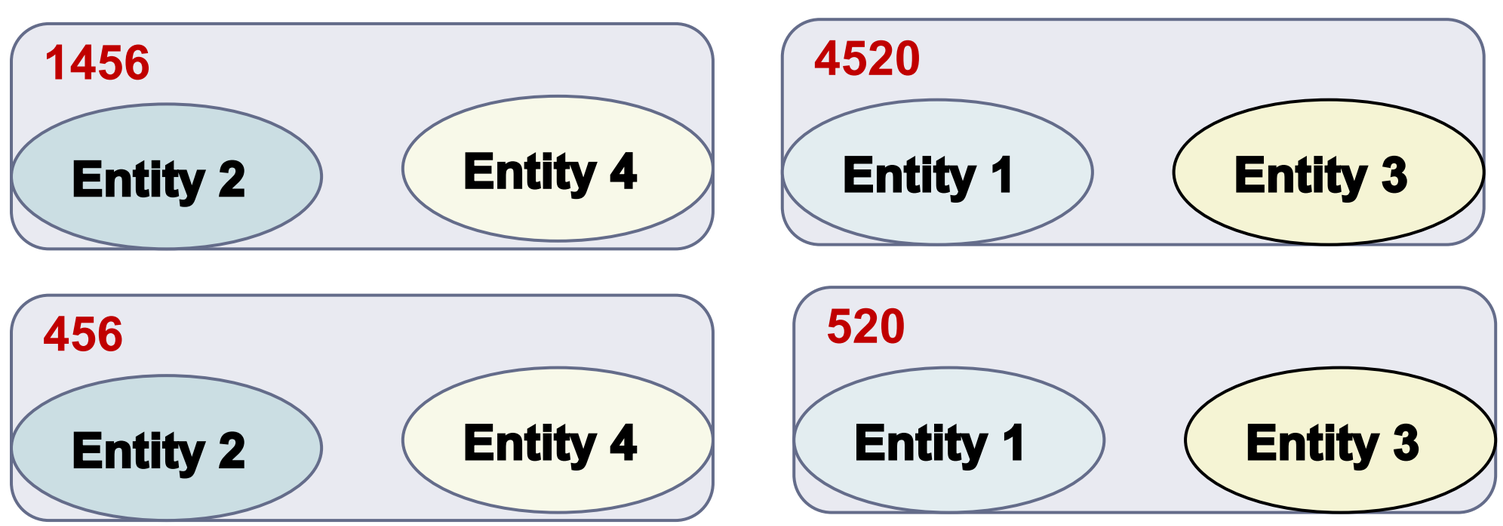
Suffix Array Blocking

[Aizawa et. al., WIRI 2005][de Vries et. al., CIKM 2009]

Converts every BKV to its list of suffixes that are longer than a predetermined minimum length l_{min}.

For l_{min} =3, the keys *91456* and *94520* yield the blocks:





Advantage:

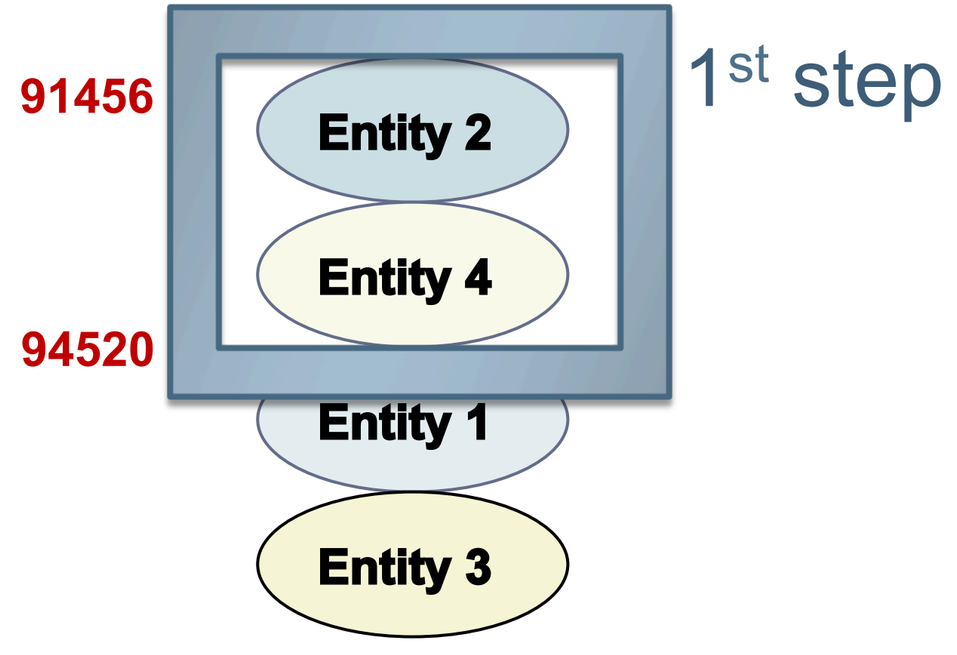
robust to noisy BKVs

Drawback:

larger blocks → higher computational cost

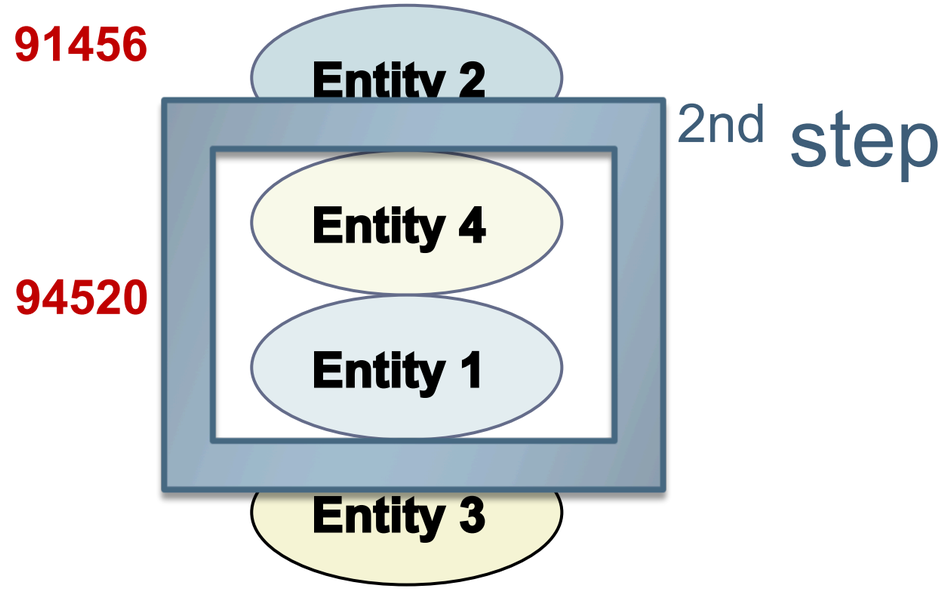
Sorted Neighborhood [Hernandez et. al., SIGMOD 1995]

1. Entities are sorted in alphabetic order of BKVs.
2. A window of fixed size slides over the sorted list.



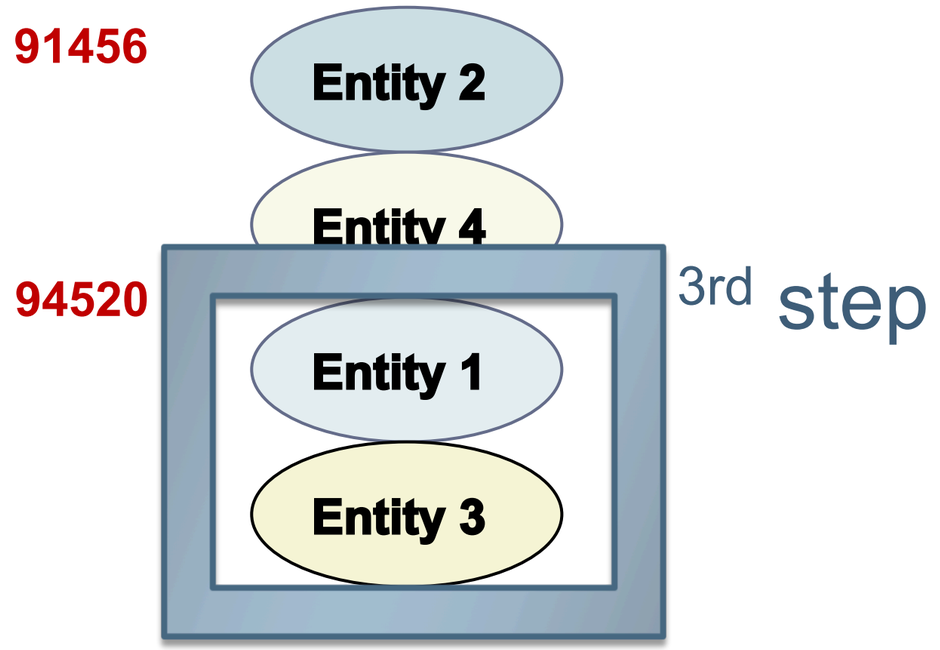
Sorted Neighborhood [Hernandez et. al., SIGMOD 1995]

1. Entities are sorted in alphabetic order of BKVs.
2. A window of fixed size slides over the sorted list.



Sorted Neighborhood [Hernandez et. al., SIGMOD 1995]

1. Entities are sorted in alphabetic order of BKVs.
2. A window of fixed size slides over the sorted list.



Drawback: the fixed window may be

- *too narrow* in some cases, leaving out matches,
- *too wide* in other cases, including many irrelevant entities.

Solution:

adaptable window size based on the similarity of BKVs (Adaptive Sorted Neighborhood [Yan et. Al., JCDL 2007])

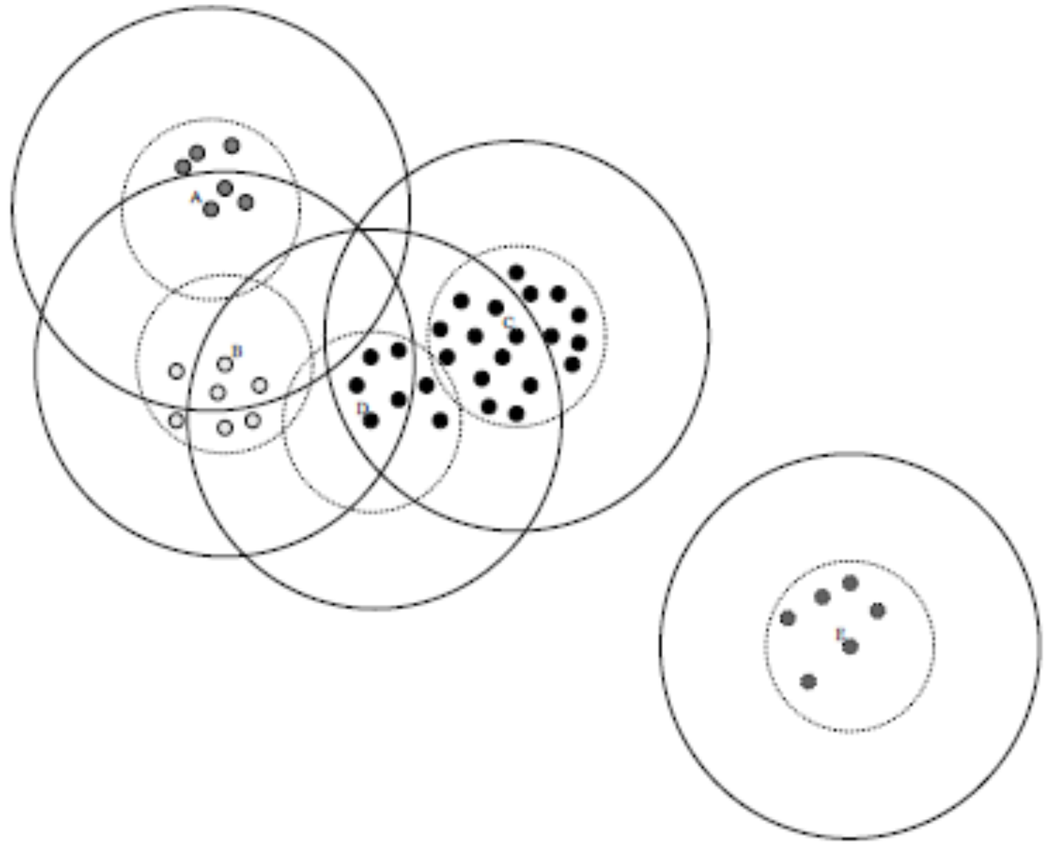
Canopy Clustering [McCallum et. al., KDD 2000]

Not based on signatures.

Instead, entity profiles are compared with a cheap string similarity metric *m* (e.g., TF-IDF, Jaccard similarity).

Algorithm:

1. Select a random entity profile e_1 from the pool of entities
2. Create a new block containing all e_i s.t. $m(e_1,e_i) < t_1$
3. Remove from the pool of entities all e_j s.t. $m(e_1,e_j) < t_2$ ($t_2 < t_1$)
4. Repeat until the pool is empty



Characteristics of Big Data

They include Web 2.0 data, Semantic Web, Dataspaces.

Voluminous, (semi-)structured datasets.

- DBPedia 3.4: 36.5 million triples and 2.1 million entities
- BTC09: 1.15 billion triples, 182 million entities.

Users are free to insert not only attribute values but also attribute names → high levels of heterogeneity.

- DBPedia 3.4: 50,000 attribute names
- Google Base: 100,000 schemata and 10,000 entity types
- BTC09: 136K attribute names

Large portion of data originating from automatic information extraction techniques → noise, tag-style values

Blocking over Big Data

Existing blocking techniques are inapplicable, due to:

- Loose schema binding.
- High levels of heterogeneity.
- Noise and missing attribute names or values.

These settings call for blocking methods that:

- Are robust to noise.
- Are decoupled from schema information.

Exemplary Challenges of Big Data

DATASET 1

Entity 1

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

Entity 2

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

DATASET 2

Entity 3

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

Entity 4

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

Loose Schema Binding

Split values

Attribute Heterogeneity

Noise

Problem Definition

Metrics for assessing block quality:

- Pair Completeness: $PC = \frac{detected\_matches}{existing\_matches}$ (recall)
- Reduction Ratio: $RR = 1 - \frac{method\_comparisons}{baseline\_comparisons}$ (savings in # of comparisons)

Problem Definition:

Given two duplicate-free entity collections (Clean-Clean ER), cluster their entities into blocks and process them so that both PC and RR are maximized.

The same applies to Dirty ER.

disclaimer:

Precision of entity matching is dependent on the entity similarity measures, and is orthogonal to the above problem.

Outline

- Introduction
- Approach
  - Core Ideas
  - Effectiveness Layer
  - Type of pair-wise comparisons
  - Metric Space for Blocking Methods
  - Meta-blocking Layer
  - Efficiency Layer
- Evaluation

Framework Outline

Three-layered approach*:

1. **Effectiveness Layer**

   Build blocks in a way that involves:

   - Attribute-agnostic functionality
   - Redundancy for robustness and high effectiveness

2. **Meta-blocking**

   Restructure a block collection into a new one with fewer comparisons, but equally high effectiveness.

2. **Efficiency Layer**

   Process blocks s.t. entire blocks or individual comparisons that do not contribute to effectiveness are discarded (thus increasing efficiency).

*Code and data are fully available at: sourceforge.net/projects/erframework/.

Effectiveness Layer

Goal:

place every pair of matching entities in at least one common block.

Solution:

- redundancy to reduce the likelihood of missed matches,
- attribute-agnostic functionality to be able to handle Big Data.

Drawbacks:

- the blocks are overlapping (i.e., repeated comparisons),
- high number of comparisons between irrelevant entities.

Three families of approaches:

1. Token Blocking
2. Attribute Clustering (Clean-Clean ER)
3. URI Semantics Blocking (Dirty ER)

Token Blocking [Papadakis et al., WSDM2011]

Attribute-agnostic blocking scheme:

- completely ignores attribute names.
- considers all attribute values.

Functionality:

1. given an entity profile, it transforms it into the set of tokens contained in its values.
2. creates one block for each distinct token → each block contains all entities with the corresponding token*.

Redundancy is inherent!

**Each block should contain at least two entities.*

Token Blocking Example

DATASET 1

Entity 1

- name=United Nations Children's Fund
- acronym=unicef
- headquarters=California

Entity 2

- name=Ann Veneman
- position=unicef
- address=California

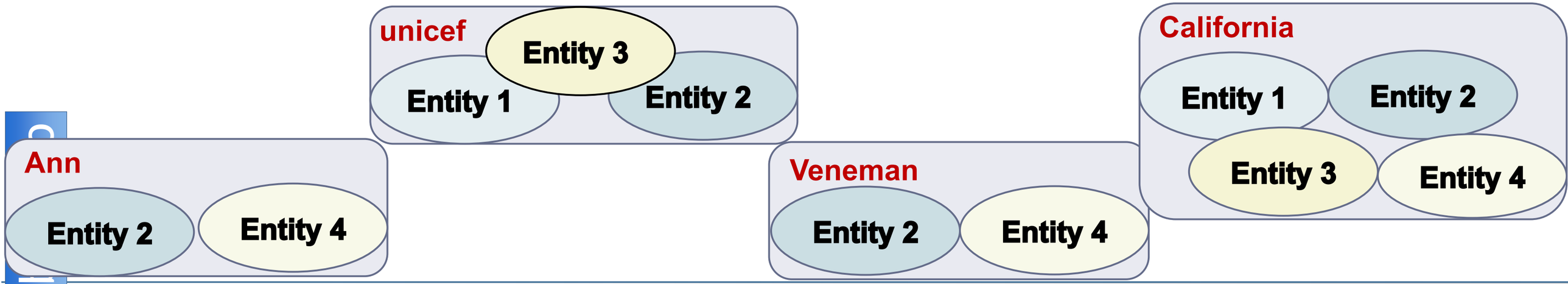
DATASET 2

Entity 3

- organization=unicef
- California
- status=active

Entity 4

- firstName=Ann
- lastName=Veneman
- residence=California



Attribute-Clustering Blocking

[Papadakis et. al., TKDE2013]

Goal

group attribute names into clusters s.t. we can apply Token Blocking independently in each cluster, yielding more efficient but equally effective blocks

Algorithm

Input data: N_1, N_2

Parameters: a string similarity metric *m*, a representation model *r*

Procedure:

for every attribute name n_i *in* N_1

find the most similar n_j *in* N_2 *s.t.* $m(r(n_i), r(n_j)) > 0$

connect them with an edge

do the same for every attribute name n_i *in* N_2

get transitive closure

get connected components

merge together singleton clusters

Attribute-Clustering Blocking – Part B

Versatile settings:

1. character n-grams with Jaccard similarity
2. token n-grams with TF-IDF
3. n-gram graphs with value similarity (graph metric)

The last one performs better, but computationally expensive.

Similar to Schema Matching, but fundamentally different:

1. Associated attribute names do not have to be semantically equivalent.
   They only have to produce good blocks.
2. All singleton attributes are associated with each other.
3. Schema matching approaches do not scale to the highly heterogeneous big data.

Evidence for URI Semantics Blocking

Attribute-agnostic blocking leads to high levels of redundancy. For Semantic Web data, three sources of evidence create blocks of lower redundancy:

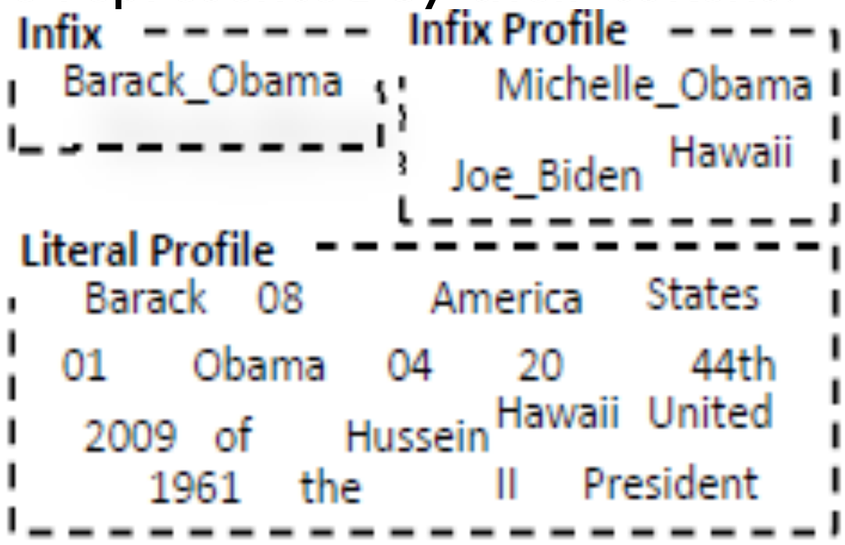
1. Semantics in entity URIs (i.e., Infix) [Papadakis et al., iiWAS 2010]
   Infixes may consist of several concatenated tokens.

Prefix	Infix	Suffix
http://dblp.l3s.de/d2r/resource/publications/books/sp/wooldridgeV99	/ThalmannN99	
http://bibsonomy.org/uri/bibtexkey/books/sp/wooldridgeV99	/ThalmannN99	/dblp

2. Relations between entities (i.e., *Infix Profile*)
   The URIs of attribute values can be represented by their Infixes.

3. Literal values (i.e., *Literal Profile*)
   Values that do not correspond to blank nodes or URIs can be represented by their tokens.

URL:	**<http://dbpedia.org/resource/Barack_Obama>**
birthname:	"Barack Hussein Obama II"
dateOfBirth:	"1961-08-04"
birthPlace:	"Hawaii" <http://dbpedia.org/resource/Hawaii>
shortDescription:	"44th President of the United States of America"
spouse:	<http://dbpedia.org/resource/Michelle_Obama>
Vicepresident:	<http://dbpedia.org/resource/Joe_Biden>



dbTrento

URI Semantics Blocking
[Papadakis et al., WSDM2012]

The above sources of evidence lead to 3 blocking methods:

1. **Infix Blocking**
   every block contains all entities whose URI has a specific Infix
2. **Infix Profile Blocking**
   every block corresponds to a specific Infix (of an attribute value) and contains all entities having it in their Infix Profile
3. **Literal Profile Blocking**
   every block corresponds to a specific token and contains all entities having it in their Literal Profile

Individually, these methods may not cover all entities
(e.g., Infix Blocking does not cover blank nodes).
However, they are complementary and can be combined
into *composite* blocking methods for higher effectiveness.

Outline

- Introduction
- Approach
  - Core Ideas
  - Effectiveness Layer
  - Type of pair-wise comparisons
  - Metric Space for Blocking Methods
  - Meta-blocking Layer
  - Efficiency Layer
- Evaluation

Meta-blocking Layer [Papadakis et. al., TKDE]

Main idea

block assignments provide valuable evidence for the similarity of entities: the more blocks two entities share, the more similar and more likely they are to be matching

Goal:

restructure a given block collection into a new one that contains substantially lower number of comparisons (RR»0), while being equally effective (same PC).

Type of pair-wise comparisons

Every comparison between entity profiles p_i and p_j belongs to one of the following types:

1. *Matching comparison* if $p_i \equiv p_j$.
2. *Redundant comparison* if p_i and p_j have been compared in a previously examined block.
3. *Superfluous comparison* if p_i or p_j or both of them have been matched to some other entity (Clean-Clean ER).
4. *Non-matching comparison* if $p_i \neq p_j$.

To enhance efficiency without any impact on effectiveness, we should discard the last three types of comparisons.

Meta-blocking Layer [Papadakis et. al., TKDE]

Main idea

block assignments provide valuable evidence for the similarity of entities: the more blocks two entities share, the more similar and more likely they are to be matching

Goal:

restructure a given block collection into a new one that contains substantially lower number of redundant and non-matching comparisons (RR»0), while being equally effective (same PC).

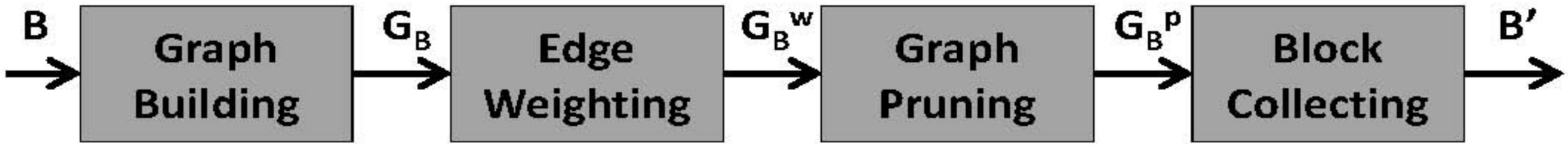
Solution: ***Blocking graph***

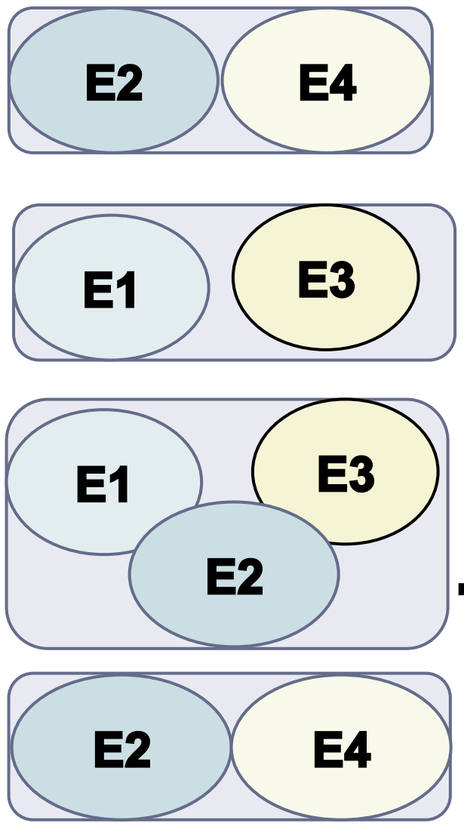
Nodes → entities

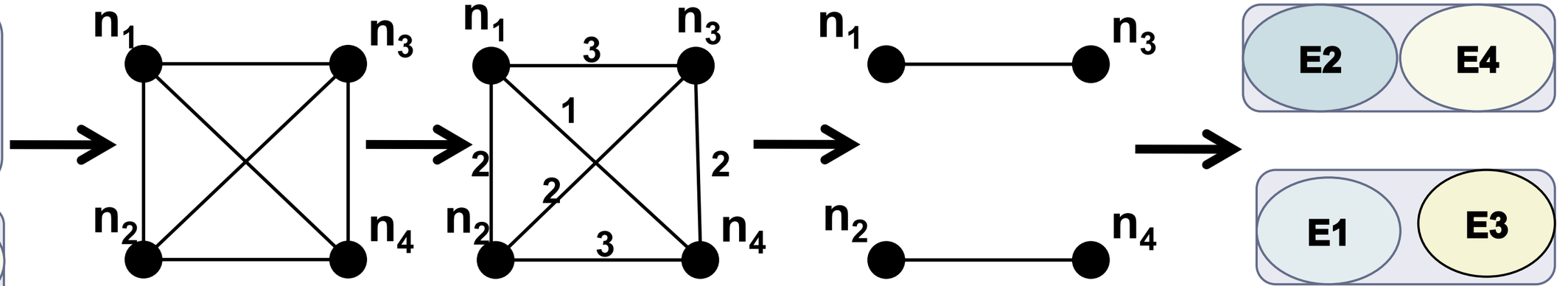
Edges → between entities co-occurring in at least one block

Weights → how similar are the adjacent entities

Outline of Meta-blocking







Metric Space for Blocking Methods

The graph pruning algorithms requires setting a threshold.
Ideally, we could set pruning thresholds using PC and RR.

However, PC and RR can only be measured *a-posteriori*:

have to first construct and examine all blocks

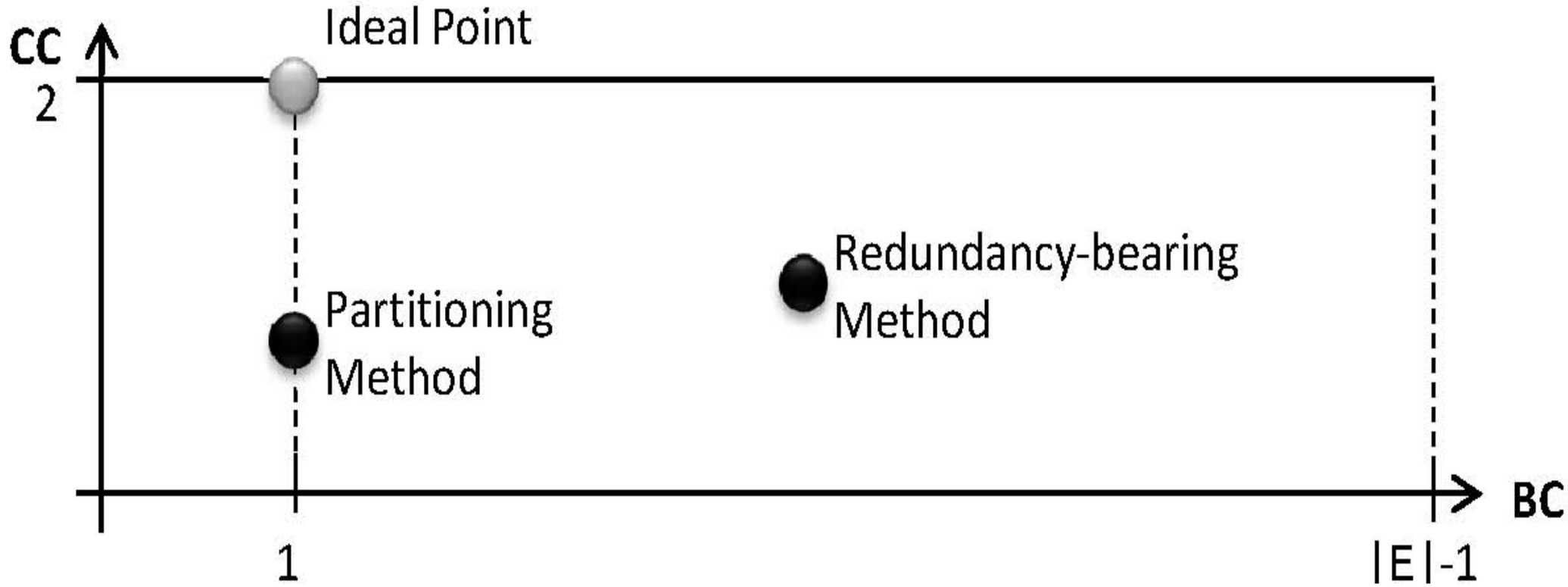
Instead, we need an *a-priori* estimation of the PC and RR values:

in order to guide the block restructuring process

without executing any comparisons

Metric Space for Blocking Methods

BC-CC metric space:



Blocking Cardinality (BC)

- average block assignments per entity: highly correlated with PC

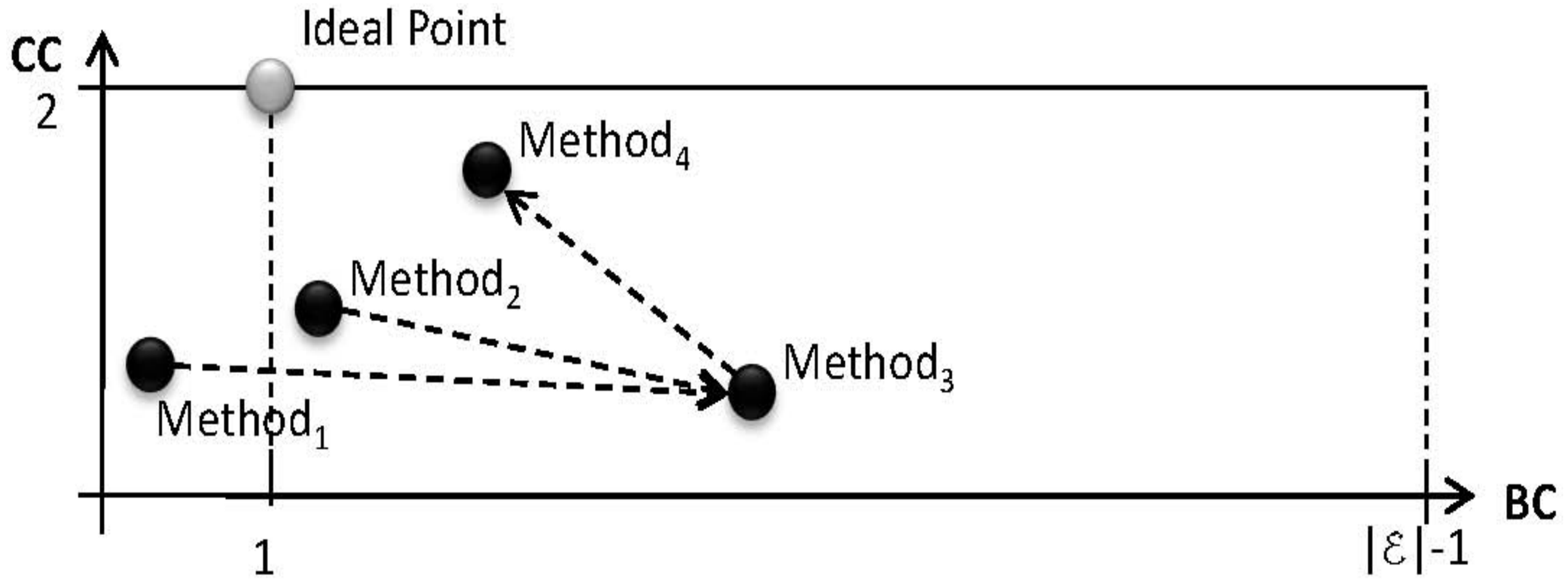
Comparisons Cardinality (CC)

- average block assignments per comparison: highly correlated with RR

Block Enhancement on BC-CC Space

Enhancing Effectiveness

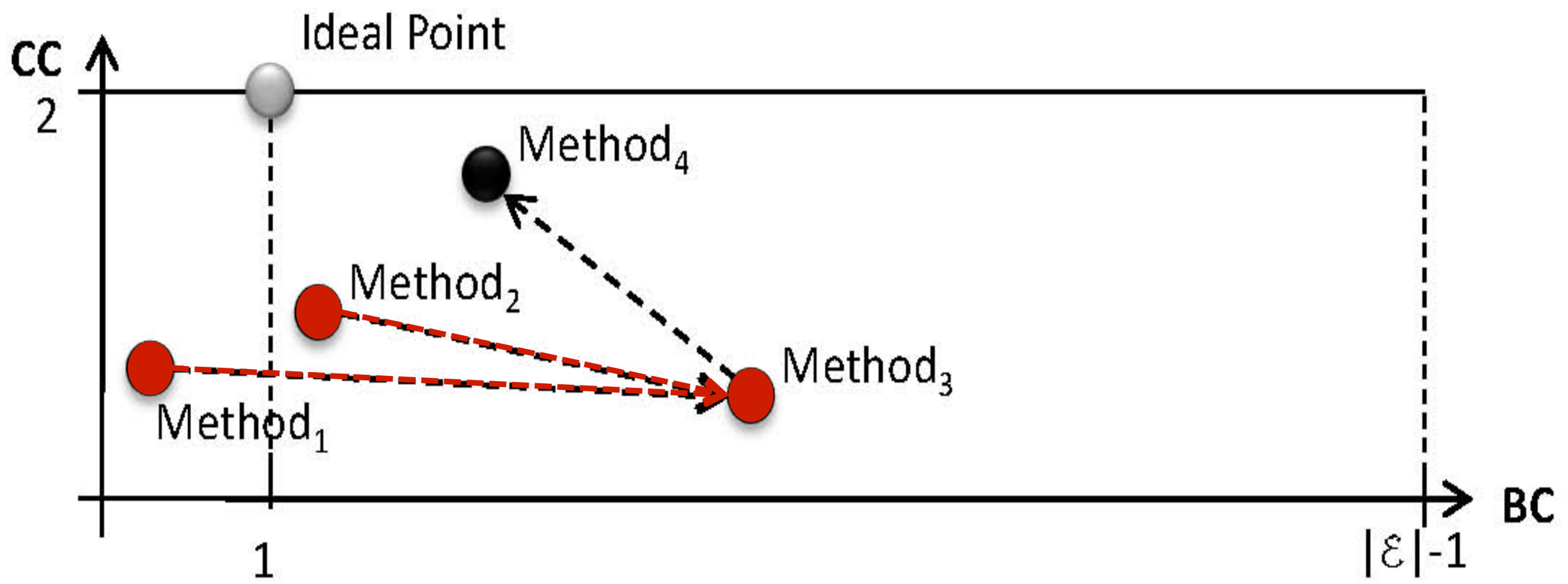
- combining complementary atomic methods into composite ones



Block Enhancement on BC-CC Space

Enhancing Effectiveness (e.g., $Method_1$, $Method_2$ → $Method_3$)

- combining complementary atomic methods into composite ones



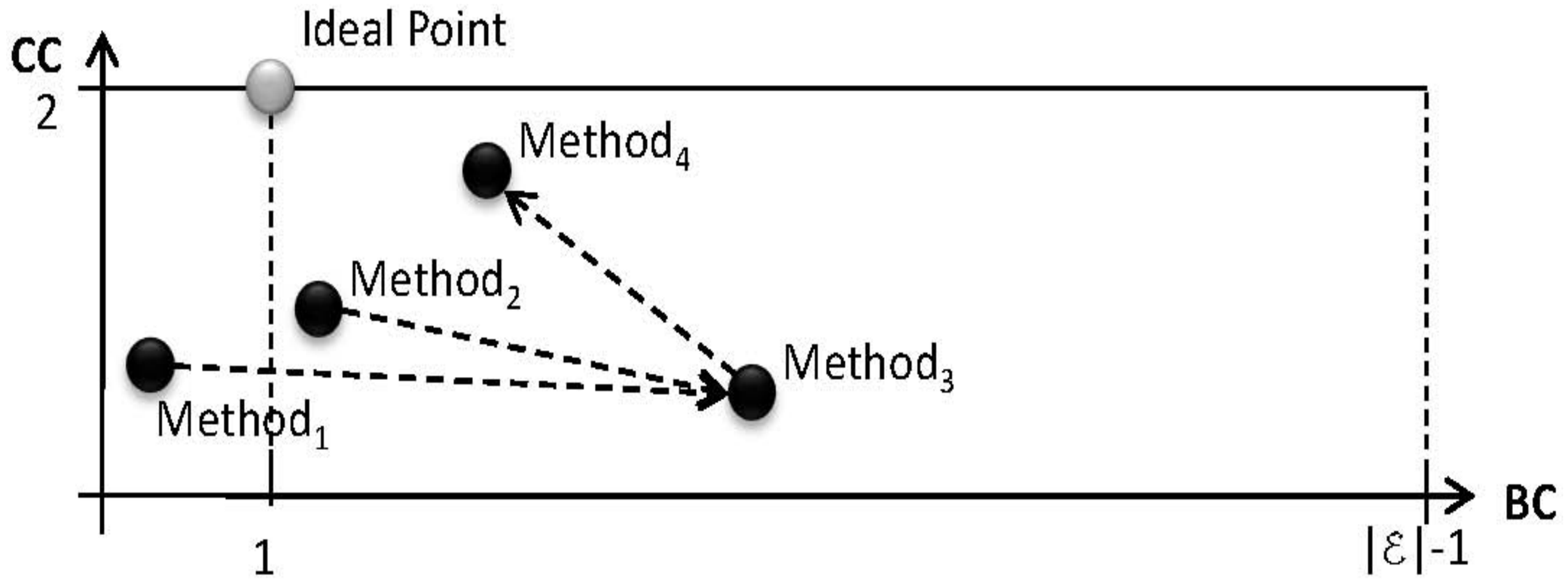
Block Enhancement on BC-CC Space

Enhancing Effectiveness (e.g., $Method_1$, $Method_2$ → $Method_3$)

- combining complementary atomic methods into composite ones

Enhancing Efficiency

- Meta-blocking
- Block Processing



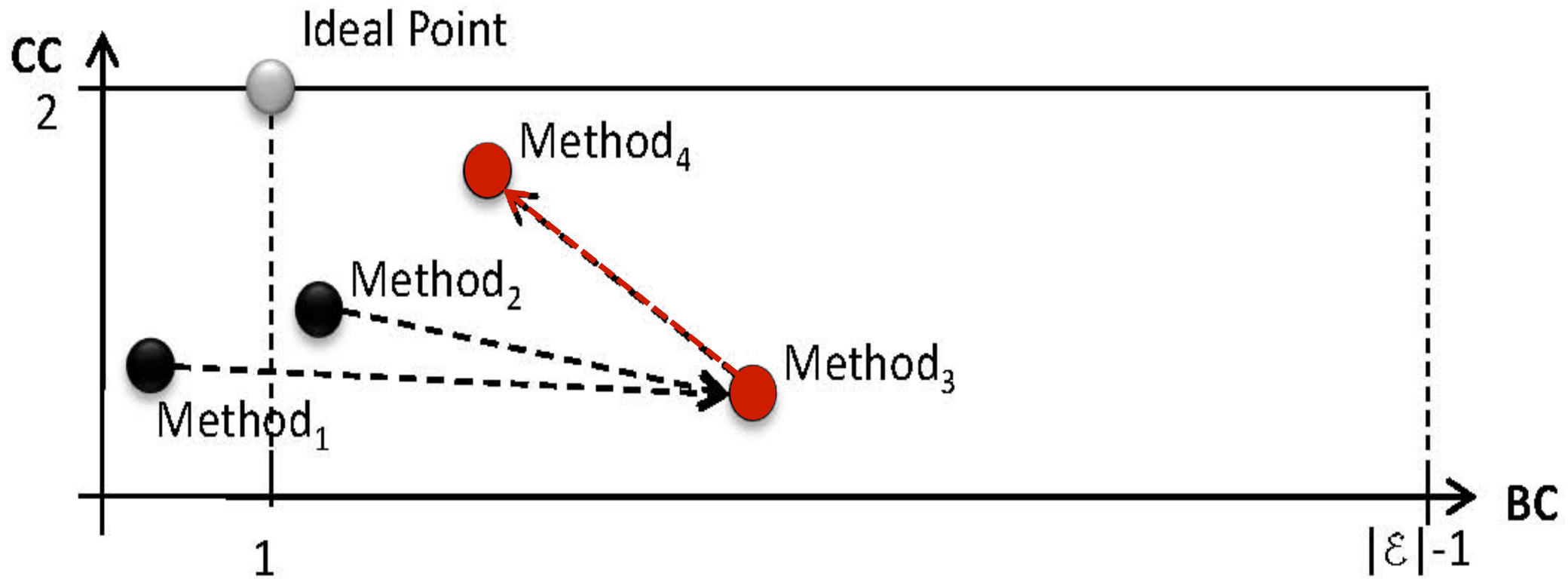
Block Enhancement on BC-CC Space

Enhancing Effectiveness (e.g., $Method_1$, $Method_2$ → $Method_3$)

- combining complementary atomic methods into composite ones

Enhancing Efficiency (e.g., $Method_3$ → $Method_4$)

- Meta-blocking
- Block Processing



Meta-blocking: Block Collecting

Transforming the pruned blocking graph into a block collection.

For undirected blocking graphs:

every retained edge creates a block of minimum size

For directed blocking graphs:

for every node (with retained *outgoing* edges), we create a new block containing the corresponding entities

Outline

- Introduction
- Approach
  - Core Ideas
  - Effectiveness Layer
  - Type of pair-wise comparisons
  - Metric Space for Blocking Methods
  - Meta-Blocking Layer
  - Efficiency Layer
- Evaluation

Efficiency Layer

Goals:

1. eliminate repeated comparisons,
2. discard superfluous comparisons,
3. avoid non-matching comparisons.

without affecting effectiveness.

Techniques:

Granularity \ Comparison's Type	Repeat Method	Superfluity Method	Non-match method	Scheduling method
Block-refinement	-	-	1. Block Purging 2. Block Pruning	Block Scheduling
Comparison-refinement	Comparison Propagation	Duplicate Propagation	Comparison Pruning	Comparison Scheduling

Block Purging

[Papadakis et al., WSDM2011] & [Papadakis et al., WSDM2012]

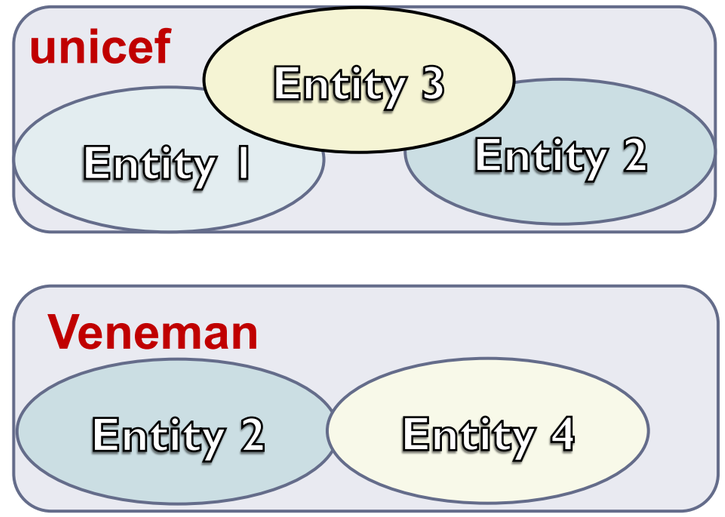
Oversized blocks: large number of comparisons, the vast majority of which is redundant, non-matching and superfluous.

Block Purging: discards oversized blocks by setting an upper limit

on size of each block [Papadakis et al., WSDM 2011], or

on number of comparisons it contains [Papadakis et al., WSDM 2012]

Purged Set of Blocks

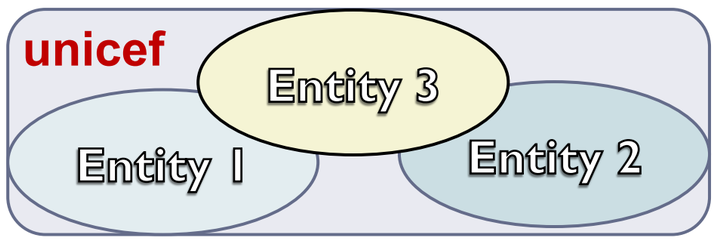


dbTrento

Duplicate Propagation
[Papadakis et al., WSDM2011]

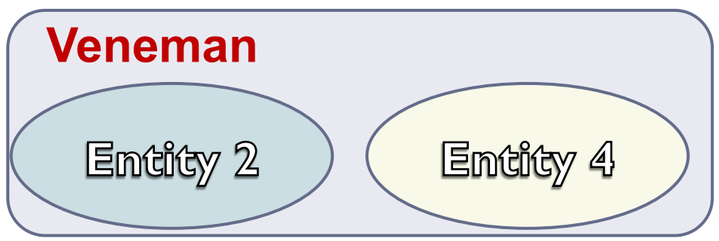
In Clean-Clean ER, matched Entities do not need to be compared again. They can be propagated through a central data structure that contains the matched duplicates from each dataset.

Before processing



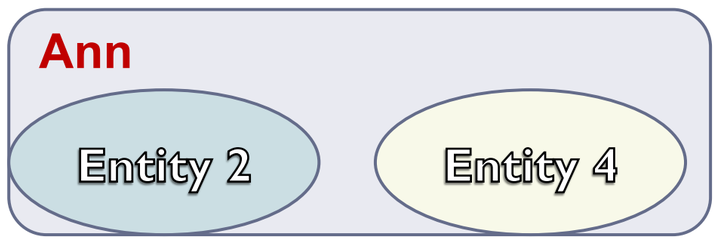
Total Comparisons: 0
Duplicates Dataset 1: {null}
Duplicates Dataset 2: {null}

After processing 1st block



Total Comparisons: 2
Duplicates Dataset 1: {Entity 1}
Duplicates Dataset 2: {Entity 3}

After processing 2nd Block



Total Comparisons: 3
Duplicates Dataset 1: {Entity 1, Entity 2}
Duplicates Dataset 2: {Entity 3, Entity 4}

After processing 3rd Block

Total Comparisons: 3!
Duplicates Dataset 1: {Entity 1, Entity 2}
Duplicates Dataset 2: {Entity 3, Entity 4}

Block Scheduling [Papadakis et al., WSDM2011]

Enhance effect of Duplicate Propagation, by detecting duplicates as early as possible:

order blocks so as to examine first those blocks that are highly likely to contain duplicates

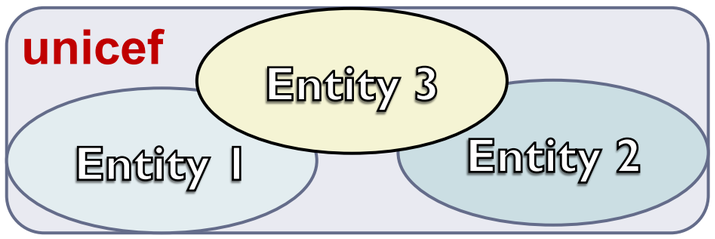
This likelihood is expressed through the utility measure that is assigned to each block:

$$U_i = \frac{gain_i}{\text{cost}_i} = \frac{1}{\max(|b_{i,1}|,|b_{i,2}|)}$$

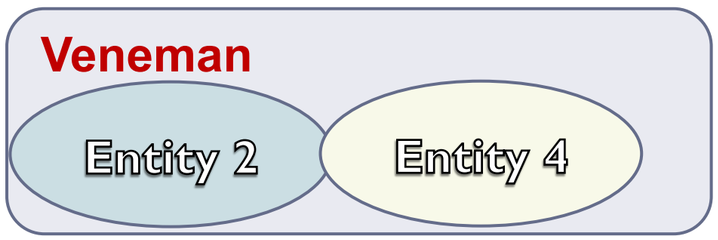
where $|b_{i,1}|$ and $|b_{i,2}|$ are number of entities from dataset 1 and 2, that are contained in block b_i

Block Scheduling Example

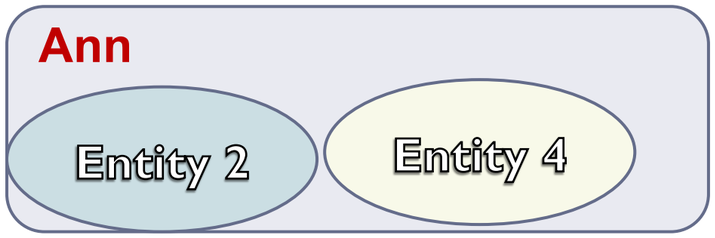
Initial Set of Blocks



$U_{unicef}=0.5$



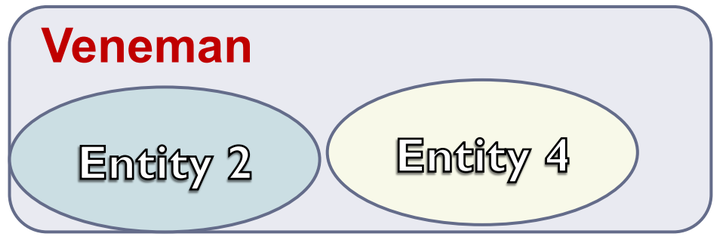
$U_{Veneman}=1$



$U_{Ann}=1$

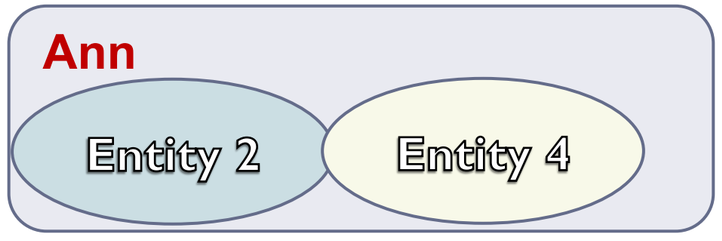
Total Comparisons: 3
(see slide 15)

Ordered Set of Blocks



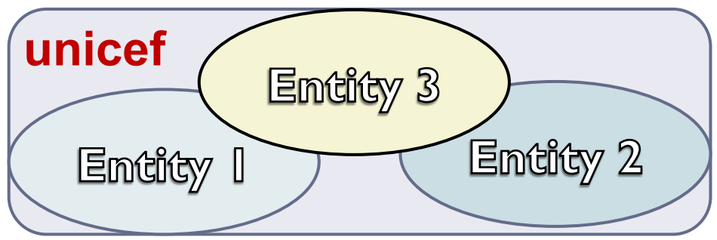
Total Comparisons: 0
Duplicates Dataset 1: {null}
Duplicates Dataset 2: {null}

$U_{Veneman}=1$



Total Comparisons: 1
Duplicates Dataset 1: {E2}
Duplicates Dataset 2: {E4}

$U_{Ann}=1$



Total Comparisons: 1
Duplicates Dataset 1: {E2}
Duplicates Dataset 2: {E4}

$U_{unicef}=0.5$

Total Comparisons: 2
Duplicates Dataset 1: {E1, E2}
Duplicates Dataset 2: {E3, E4}

Effect of Block Scheduling

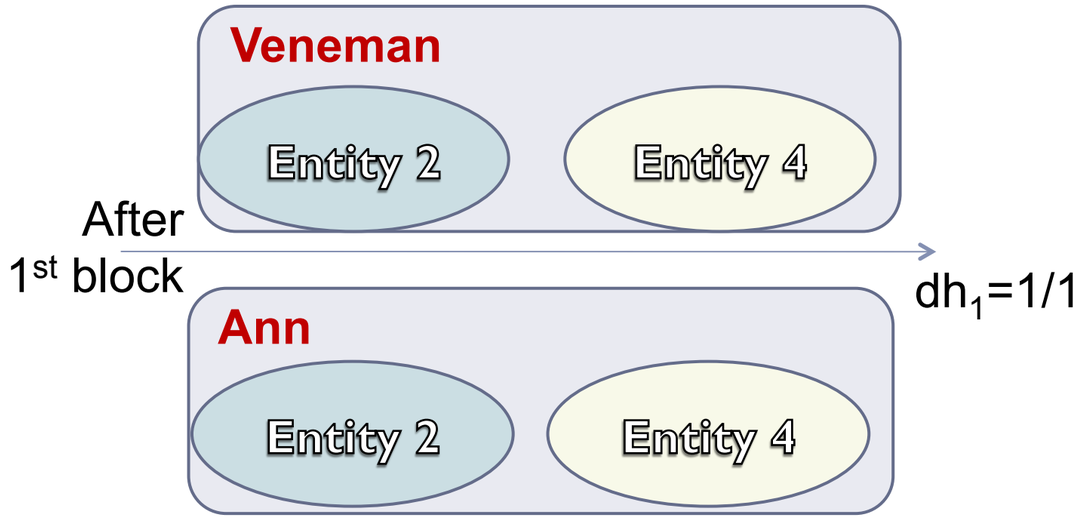
cost of identifying **new** duplicates is called *duplicate overhead (dh)*: $dh_k = \frac{comparisons_k}{duplicates_k}$

comparisons$_k$: # of comparisons executed after examining the (k-1)-th block containing new duplicates

duplicates$_k$: # of matched pairs of entities in the *k-th* block <u>containing new duplicates.</u>

dh *increases* as we move from first block to next ones:

due to Block Scheduling, the lower the execution order of a block is, the more comparisons and the less unique duplicates it entails



Block Pruning [Papadakis et al., WSDM 2011]

dh provides a good estimation of the duplicates that remain to be detected → the higher its value is, the more unlikely it is to identify new duplicates in the remaining blocks at a reasonable cost (i.e., #comparisons).

Therefore, the ER process can be terminated prematurely, when dh gets high.

Block Pruning sets an upper limit on duplicate overhead, called dh_{max}, based on the total number of comparisons *n* contained in the original set of blocks: $dh_{\max} = 10^{\log n/2}$

As soon as dh exceeds dh_{max}, the entire process is terminated.

Comparison Propagation

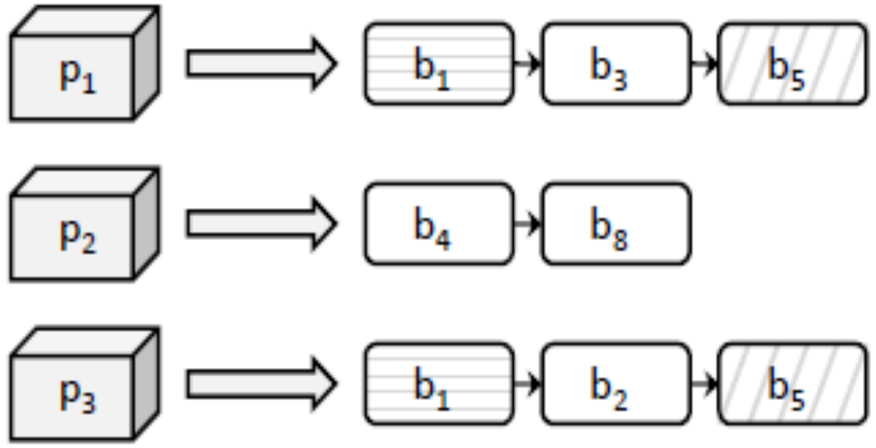
[Papadakis et al., JCDL2011]

Goal : eliminate all redundant comparisons.

Naïve solution: store all executed comparisons in a hash table → not scalable

Our approach: indirect propagation of all executed comparisons in three steps

1. **Block Enumeration**: assign to each block an index denoting its position on the block processing list.
2. **Inverted Index**: Data structure associating entities with block indices.



3. Two entities are compared iff they satisfy the **Least Common Block Index Condition**: the index of the current block is their minimum common block index. E.g., p_1 and p_3 are compared in b_1, but the condition does not hold in b_5.

Comparison Pruning [Papadakis et al., SWIM 2011]

Goal: discard non-matching comparisons.

How can we infer that two entities are non-matching without actually comparing them?

Examine number of blocks they have in common!

Comparison Pruning discards a comparison between p_i and p_j if it does not satisfy the following condition:

$$\mathbf{JS_{min} \leq ES(p_i, p_j)}$$

where:

$JS(p_i, p_j)$ is the Jaccard coefficient of their block lists, and

$$ES_{min} = \frac{a \cdot min(iBC_{t,c}^{\mathcal{E}_1}, iBC_{t,c}^{\mathcal{E}_2})}{iBC_{t,c}^{\mathcal{E}_1} + iBC_{t,c}^{\mathcal{E}_2} - a \cdot min(iBC_{t,c}^{\mathcal{E}_1}, iBC_{t,c}^{\mathcal{E}_2})}$$

Comparison Scheduling
[Papadakis et. al., TKDE1]

Enhances the effect of Duplicate Propagation: it schedules the execution of individual comparisons so that those involving real matches are executed first.

The utility of every comparison between p_i and p_j is:

$$u(p_i,p_j) = JS(p_i, p_j) \cdot ICF(p_i) \cdot ICF(p_j)$$

where $JS(p_i, p_j)$ is the Jaccard coefficient of the block lists, and $ICF(p_k)$ is the inverse comparison frequency of p_k.

To reduce its high computational cost, it is usually applied after Comparison Propagation and Comparison Pruning.

Composing ER workflows

[Papadakis et. al., TKDE2013]

Block Processing

1st step	2nd step	3rd step	4th step	5th step	6th step
Block Building	**Meta-blocking**	**Core Methods**	**Scheduling Methods**	**Repeat Methods**	**Non-match Methods**
Token Blocking OR **Agnostic Clustering** OR **Total Description**	**WEP** OR **CEP** OR **WNP** **OR** CNP	**Block Purging** AND **Duplicate Propagation**	**Block Scheduling** OR **Comparison Scheduling**	**Comparison Propagation**	**Block Pruning** OR **Comparison Pruning**

Actual selection depends on the application requirements and the available resources.

In general, comparison-refinement methods are more accurate, but more time- and resource-consuming than block-refinement.

Outline

- Introduction
- Approach
- Evaluation
  - Experimental Settings
  - Block Building Performance
  - Scalability of Token Blocking
  - Block Purging Performance
  - Meta-blocking Performance
  - Block Processing Performance

Experimental Settings

Metrics

- Pair Completeness: $PC = \frac{detected\_matches}{existing\_matches}$ (recall)
- Reduction Ratio: $RR = 1 - \frac{method\_comparisons}{baseline\_comparisons}$ (savings in # of comparisons)

Datasets

	BTC09 (Dirty ER)		Dbpedia (Clean-Clean ER)	
			30RC	3.4
Entities	182 million		1,190,734	2,164,058
Name-Value Pairs	1.15 billion		17,453,516	36,653,387
Duplicates	11,591 (IFP)	5,988,554 (Same-As)	892,586	

Note 1: out of 43,75 million distinct triples of $D_{DBPedia}$, only 10,36 million (<25%) are common.

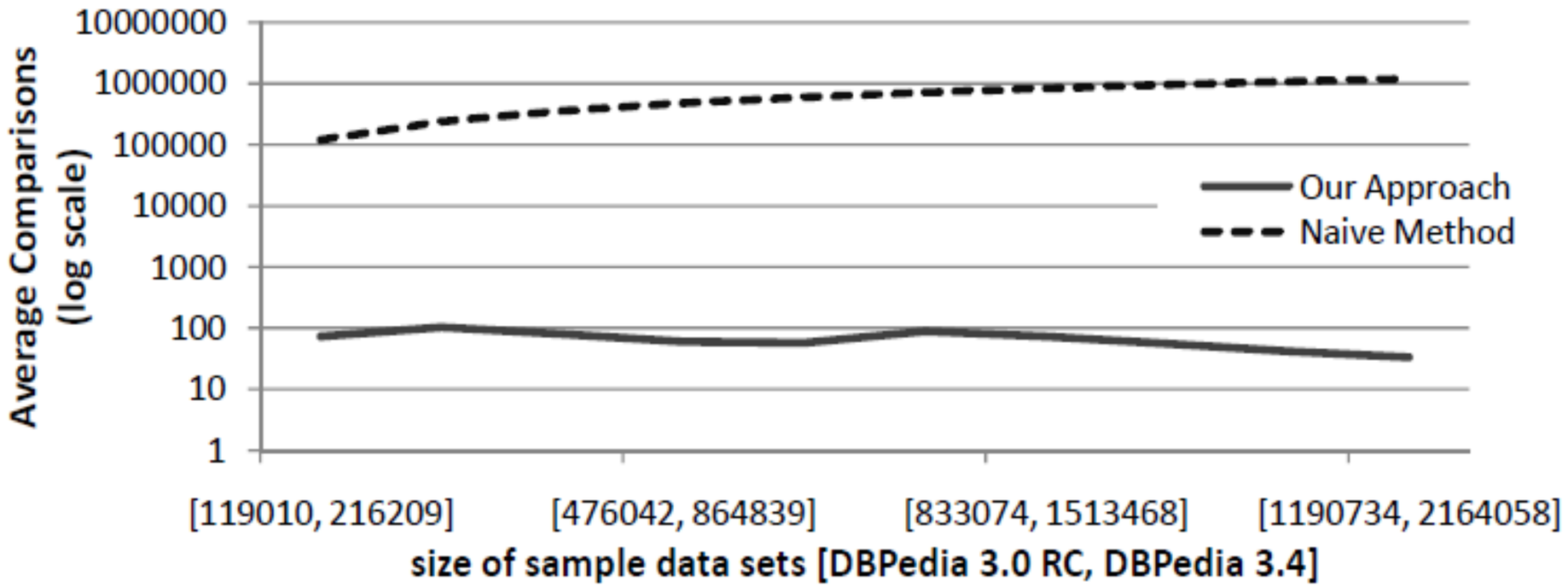
Note 2: IFP corresponds to pairs of matching entities that are **implicitly** denoted via their common values for some Inverse Functional Properties. Same-As corresponds to pairs of matching entities that are **explicitly** denoted through same-as statements.

Scalability of Token Blocking

We considered subsets of $D_{DBpedia}$.

For each subset we estimated the following metric:

$$\text{comparisons\_per\_entity} = \frac{\text{average\_comparisons}}{|\text{DBPedia}_{3.0rc}| + |\text{DBPedia}_{3.4}|}$$



3-4.5 orders of magnitude less comparisons (efficiency)

PC always above 95% (recall)

Block Building Performance

DBPedia	Comparisons	Duplicates	PC	RR
Token Blocking	$6.18 \cdot 10^{12}$	892,560	99.99%	-
Term Vector EM	$6.38 \cdot 10^{12}$	891,546	99.99%	-
Term Vector AC	$6.18 \cdot 10^{12}$	891,560	99.99%	0.01%
Trigrams AC	$1.05 \cdot 10^{12}$	892,425	99.98%	83.04%
Trigram Graphs AC	**$1.03 \cdot 10^{12}$**	**892,516**	**99.99%**	**83.39%**

BTC09	Comparisons	PC IFP	PC SameAs	RR
Token Blocking	$2.59 \cdot 10^{16}$	99.32%	92.26%	-
Infix Blocking	$6.24 \cdot 10^{12}$	59.31%	49.60%	99.98%
Infix Profile Blocking	$1.07 \cdot 10^{15}$	84.19%	99,99%	95.86%
Literal Profile Blocking	$2.67 \cdot 10^{15}$	94.49%	24.51%	89.68%
Infix+Infix Profile	$1.10 \cdot 10^{15}$	85.16%	87.14%	95.77%
Infix+Literal Profile	$3.25 \cdot 10^{15}$	96.43%	65.67%	87.46%
Infix Profile+Literal Profile	$3.81 \cdot 10^{15}$	96.64%	52.09%	85.30%
All (Combined)	**$3.90 \cdot 10^{15}$**	**97.98%**	**91.13%**	**84.96%**

Block Purging Performance

DBPedia	Comparisons	Duplicates	PC	RR
Token Blocking	$5.68 \cdot 10^{10}$	891,767	99.91%	99.08%
Term Vector EM	$4.34 \cdot 10^{10}$	891,709	99.90%	99.32%
Term Vector AC	$5.68 \cdot 10^{10}$	891,767	99.91%	99.08%
Trigrams AC	$3.06 \cdot 10^{10}$	892,402	99.98%	97.08%
Trigram Graphs AC	**$2.42 \cdot 10^{10}$**	**892,463**	**99.99%**	**97.64%**

BTC09	Comparisons	PC IFP	PC SameAs	RR
Token Blocking	$15.49 \cdot 10^{11}$	96.79%	60.52%	99.99%
Infix Blocking	$0.004 \cdot 10^{11}$	58.85%	49.54%	99.99%
Infix Profile Blocking	$1.62 \cdot 10^{11}$	76.53%	41.36%	99.99%
Literal Profile Blocking	$9.37 \cdot 10^{11}$	94.34%	18.29%	99.99%
Infix+Infix Profile	$1.69 \cdot 10^{11}$	72.64%	86.60%	99.99%
Infix+Literal Profile	$8.79 \cdot 10^{11}$	94.48%	63.75%	99.99%
Infix Profile+Literal Profile	$12.00 \cdot 10^{11}$	93.57%	50.03%	99.99%
All (Combined)	**$11.64 \cdot 10^{11}$**	**95.37%**	**89.35%**	**99.99%**

Block Processing Performance

		Comparisons	RR	Duplicates	PC	Time
	Block Purging	$2.42 \cdot 10^{10}$	-	892,463	99.99%	0.05
WF_1	Block Scheduling	$1.55 \cdot 10^{10}$	93.58%	892,463	99.99%	0.16
	Block Pruning	$7.24 \cdot 10^{7}$	99.70%	879,446	98.53%	0.01
WF_2	Comparison Propagation	$1.24 \cdot 10^{10}$	48.86%	892,463	99.99%	5.75
	Block Scheduling	$9.20 \cdot 10^{8}$	96.20%	892,463	99.99%	0.16
	Comparison Pruning	$4.98 \cdot 10^{7}$	99.79%	837,286	93.80%	4.14
WF_3	Comparison Propagation	$1.24 \cdot 10^{10}$	48.86%	892,463	99.99%	5.75
	Comparison Pruning	$4.32 \cdot 10^{8}$	98.21%	837,286	93.80%	4.14
	Comparison Scheduling	$4.46 \cdot 10^{7}$	99.82%	837,286	93.80%	0.51

Meta-blocking Performance

		Comparisons	RR	PC	ΔPC
	Token Blocking	$3.98 \cdot 10^{10}$	-	99.91%	-
WEP	ARCS	$2.85 \cdot 10^{8}$	99.28%	92.45%	-7.45%
	CBS	$3.40 \cdot 10^{9}$	91.46%	95.47%	-4.42%
	ECBS	$5.77 \cdot 10^{9}$	85.50%	99.66%	-0.23%
	JS	$1.11 \cdot 10^{10}$	71.80%	99.73%	-0.16%
	EJS	$1.10 \cdot 10^{10}$	72.32%	99.77%	-0.11%
CEP	ARCS	$1.85 \cdot 10^{9}$	95.34%	99.41%	-0.48%
	CBS	$3.57 \cdot 10^{9}$	91.04%	99.35%	-0.54%
	ECBS	$9.94 \cdot 10^{9}$	75.02%	99.75%	-0.14%
	JS	$1.96 \cdot 10^{10}$	50.76%	99.87%	-0.02%
	EJS	$1.99 \cdot 10^{10}$	49.74%	99.88%	-0.01%

	Comp.	RR	PC				
			ARCS	CBS	ECBS	JS	EJS
WNP	$0.26 \cdot 10^{8}$	99.94%	79.46%	51.71%	61.14%	82.09%	79.61%
CNP	$0.50 \cdot 10^{8}$	99.88%	93.43%	92.35%	94.05%	95.57%	95.99%

Meta-blocking Time Requirements

Materialization Time (***MT***): time for Graph Building and Edge Weighting.

Restructure Time (***RT***): time for Graph Pruning and Block Collecting.

Comparisons Time (***CT***): time for executing the retained comparisons.

Performance over DBPedia in hours, using Intel Xeon E5472 3.0 GHz and 16GB of RAM. Profile comparison was done with Jaccard similarity.

		MT	RT	CT	Total
Token Bl.		0	0	142.37	142.37
WEP	ARCS	7.48	17.00	25.13	49.62
	CBS	8.08	8.53	26.81	43.42
	ECBS	11.69	9.40	27.93	49.03
	JS	7.74	6.75	26.76	41.26
	EJS	6.87	7.12	32.46	46.45
CEP	ARCS	6.87	16.08	1.81	24.76
	CBS	7.12	6.97	4.64	18.73
	ECBS	7.26	7.40	2.84	17.49
	JS	6.81	6.56	0.28	13.66
	EJS	6.81	7.45	0.25	14.51
WNP	ARCS	6.81	17.58	13.05	40.64
	CBS	6.84	6.87	29.08	49.99
	ECBS	6.84	7.08	35.56	56.84
	JS	7.36	9.07	26.74	42.24
	EJS	6.81	7.44	36.85	58.83
CNP	ARCS	6.81	17.58	3.49	27.89
	CBS	6.84	6.87	5.35	19.06
	ECBS	6.84	7.08	2.05	15.97
	JS	7.36	9.07	0.78	17.21
	EJS	6.81	7.44	0.85	15.10

Conclusions

Entity resolution (ER) is an increasingly important problem

- Becoming more relevant in the big data era

Proposed a framework for ER in big data

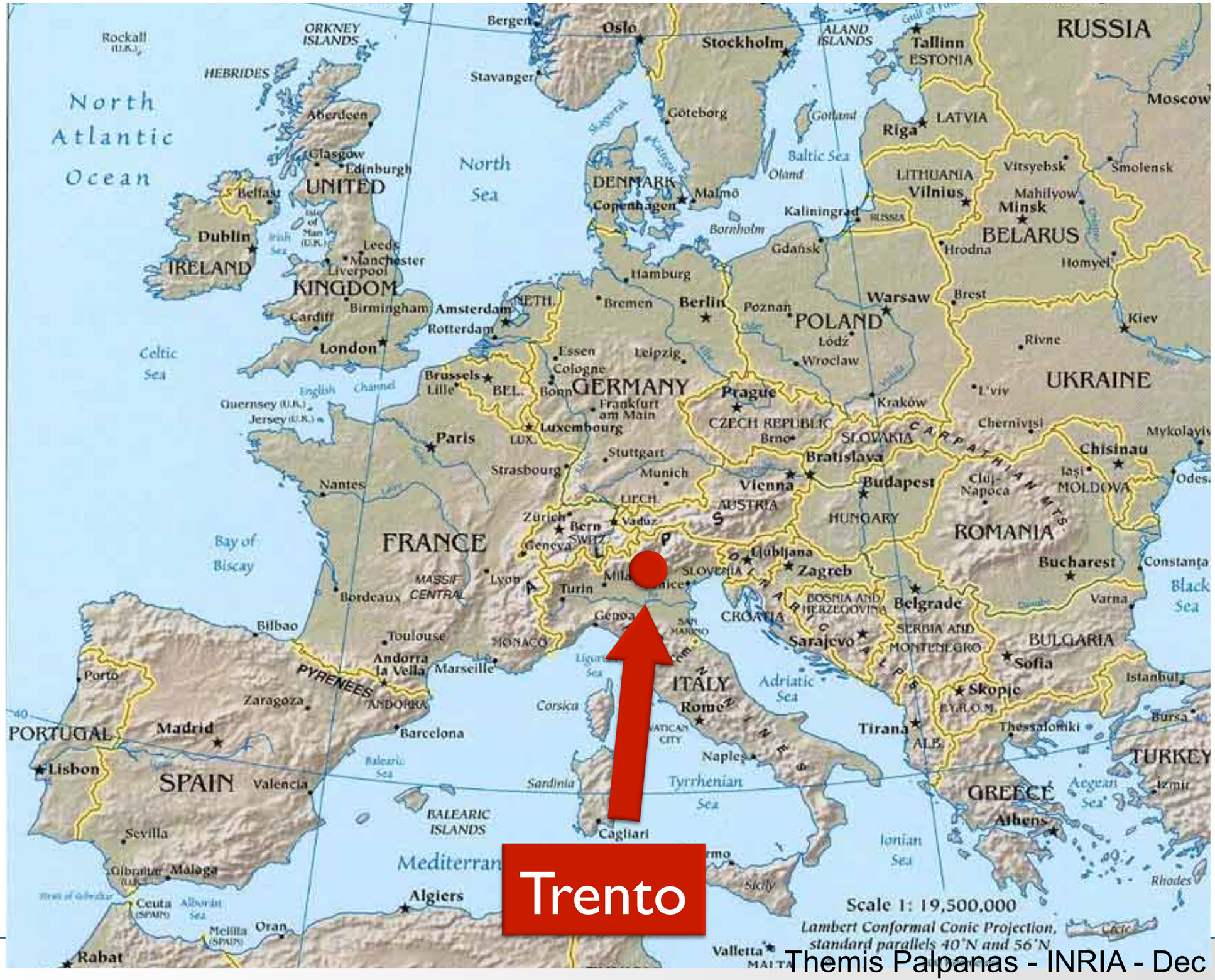
- Systematic study and organization of all proposed techniques
  - effectiveness layer, meta-blocking, efficiency layer
- Novel techniques targeted to big data scenarios
- Lead (up) to more than 95% RR for more than 99% PC!

Several challenges ahead

- Scalability is always a goal
- Incremental methods are necessary

collaborations!

Trento, Italy



dbTrento

dbTrento: http://db.disi.unitn.eu

dbTrento

activities described in invited paper:

SIGMOD Record, Sep 2012

dbTrento: http://db.disi.unitn.eu

thank you!

References – Part A

[Aizawa et. al., WIRI 2005] Akiko N. Aizawa, Keizo Oyama, "A Fast Linkage Detection Scheme for Multi-Source Information Integration" in WIRI, 2005.

[Baxter et. al., KDD 2003] R. Baxter, P. Christen, T. Churches, "A comparison of fast blocking methods for record linkage", in Workshop on Data Cleaning, Record Linkage and Object Consolidation at KDD, 2003.

[Bilenko et. al., ICDM 2006] M. Bilenko, B. Kamath, R. Mooney, "Adaptive blocking: Learning to scale up record linkage", in ICDM, 2006.

[Christen, TKDE 2011] P. Christen, " A survey of indexing techniques for scalable record linkage and deduplication." in TKDE 2011.

[de Vries et. al., CIKM 2009] T. de Vries, H. Ke, S. Chawla, P. Christen, "Robust record linkage blocking using sux arrays", in CIKM, 2009.

[Gravano et. al., VLDB 2001] L. Gravano, P. Ipeirotis, H. Jagadish, N. Koudas, S. Muthukrishnan, D. Srivastava, "Approximate string joins in a database (almost) for free', in VLDB, 2001.

[Hernandez et. al., SIGMOD 1995] M. Hernandez, S. Stolfo, "The merge/purge problem for large databases", in SIGMOD, 1995.

References – Part B

[Jin et. al., DASFAA 2003] L. Jin, C. Li, S. Mehrotra, "Efficient record linkage in large data sets", in DASFAA, 2003.

[Kim et. al., EDBT 2010] H. Kim, D. Lee, "HARRA: fast iterative hashed record linkage for large-scale data collections", in EDBT, 2010.

[Madhavan et. al., CIDR 2007] J. Madhavan, S. Cohen, X. Dong, A. Halevy, S. Jeffery, D. Ko, C. Yu, "Web-scale data integration: You can afford to pay as you go", in CIDR, 2007

[McCallum et. al., KDD 2000] A. McCallum, K. Nigam, L. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching", in KDD, 2000.

[Michelson et. al., AAAI 2006] M. Michelson, C. Knoblock, "Learning blocking schemes for record linkage", in AAAI, 2006.

[Papadakis et al., iiWAS 2010] G. Papadakis, G. Demartini, P. Fankhauser, P. Karger, "The missing links: discovering hidden same-as links among a billion of triples", in iiWAS 2010.

[Papadakis et al., WSDM 2011] G. Papadakis, E. Ioannou, C. Niederee, P. Fankhauser, "Efficient entity resolution for large heterogeneous information spaces", in WSDM 2011.

References – Part C

[Papadakis et al., JCDL 2011] G. Papadakis, E. Ioannou, C. Niederee, T. Palpanas, W. Nejdl, "Eliminating the redundancy in blocking-based entity resolution methods", in JCDL 2011.

[Papadakis et al., SWIM 2011] G. Papadakis, E. Ioannou, C. Niederee, T. Palpanas, W. Nejdl, "To Compare or Not to Compare: making Entity Resolution more Efficient", in SWIM workshop (collocated with SIGMOD), 2011.

[Papadakis et al., WSDM 2012] G. Papadakis, E. Ioannou, C. Niederee, T. Palpanas, W. Nejdl, "Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data", in WSDM 2012.

[Papadakis et. al., TKDE2013] George Papadakis, Ekaterini Ioannou, Themis Palpanas, Claudia Niederee, Wolfgang Nejdl, "A Blocking Framework for Entity Resolution in Highly Heterogeneous Information Spaces", in IEEE TKDE (to appear).

[Papadakis et. al., TKDE] George Papadakis, Georgia Koutrika, Themis Palpanas, Wolfgang Nejdl, "Meta-Blocking: Taking Entity Resolution to the Next Level", in IEEE TKDE (currently under revision).

[Yan et. Al., JCDL 2007] Su Yan, Dongwon Lee, Min-Yen Kan, C. Lee Giles, "Adaptive sorted neighborhood methods for efficient record linkage", in JCDL 2007.