updated: 12 May 2017

Blocking for BIG Data Integration

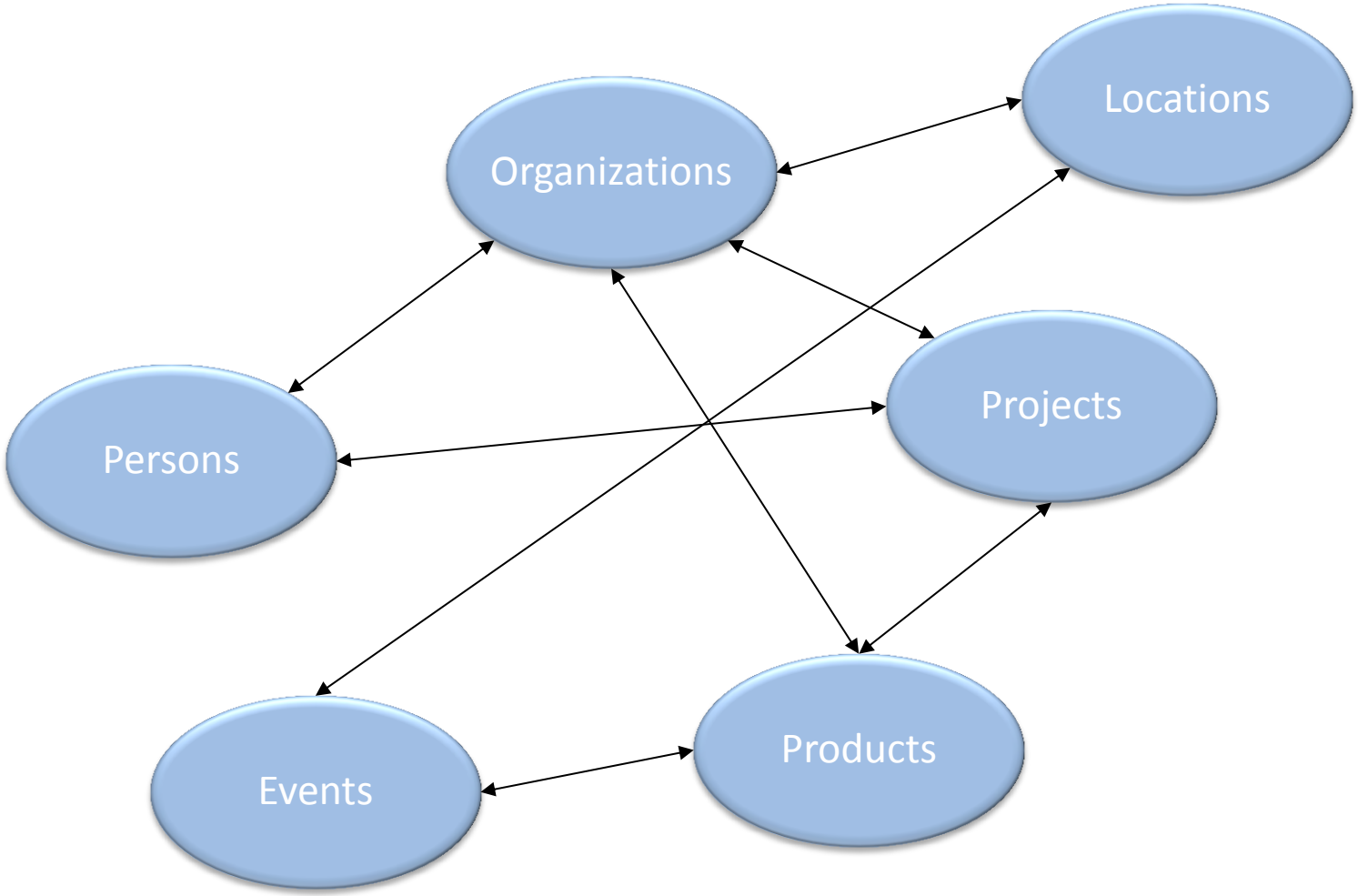
Challenges, Algorithms, Practical Examples

George Papadakis
University of Athens
gpapadis@di.uoa.gr

Themis Palpanas
Paris Descartes University
themis@mi.parisdescartes.fr

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 real-world "entity"?

However …

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

London 런던 ܠܘܢܕܘܢ लंडन लंदन લંડન ለንደን ロンドン লন্ডন ลอนดอน இலண்டன் ლონდონი Llundain Londain Londe Londen Londen Londen Londinium London Londona Londonas Londoni Londono Londra Londres Londrez Londyn Lontoo Loundres Luân Đôn Lunden Lundúnir Lunnainn Lunnon لندن لندن لندن لوندون לאנדאן לונדון Λονδίνο Лёндан Лондан Лондон Лондон Лондон Լոնդոն 伦敦 …

However …

How many names, descriptions or IDs (URIs) are used for the same real-world “entity”?

London 런던 ལོན་ཊོན लंडन लंदन લંડન ለንደን ロンドン লন্ডন ลอนดอน இலண்டன் ლონდონი Llundain Londain Londe Londen Londen Londen Londinium London Londona Londonas Londoni Londono Londra Londres Londrez Londyn Lontoo Loundres Luân Đôn Lunden Lundúnir Lunnainn Lunnon لندن لندن لندن لوندون לונדון לאנדאן Λονδίνο Лёндан Лондан 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, …

However …

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

London 런던 لندن লন্ডন լոնդոն ለንደን ロンドン లండన ಲಂಡನ ลอนดอน இலண்டன் ლონდონი Llundain Londain Londe Londen Londen Londen Londinium London Londona Londonas Londoni Londono Londra Londres Londrez Londyn Lontoo Loundres Luân Đôn Lunden Lundúnir Lunnainn Lunnon لندن لندن لندن لوندون לאנדאן לונדון Λονδίνο Лёндан Лондан 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
…

… 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
- ...

… or …

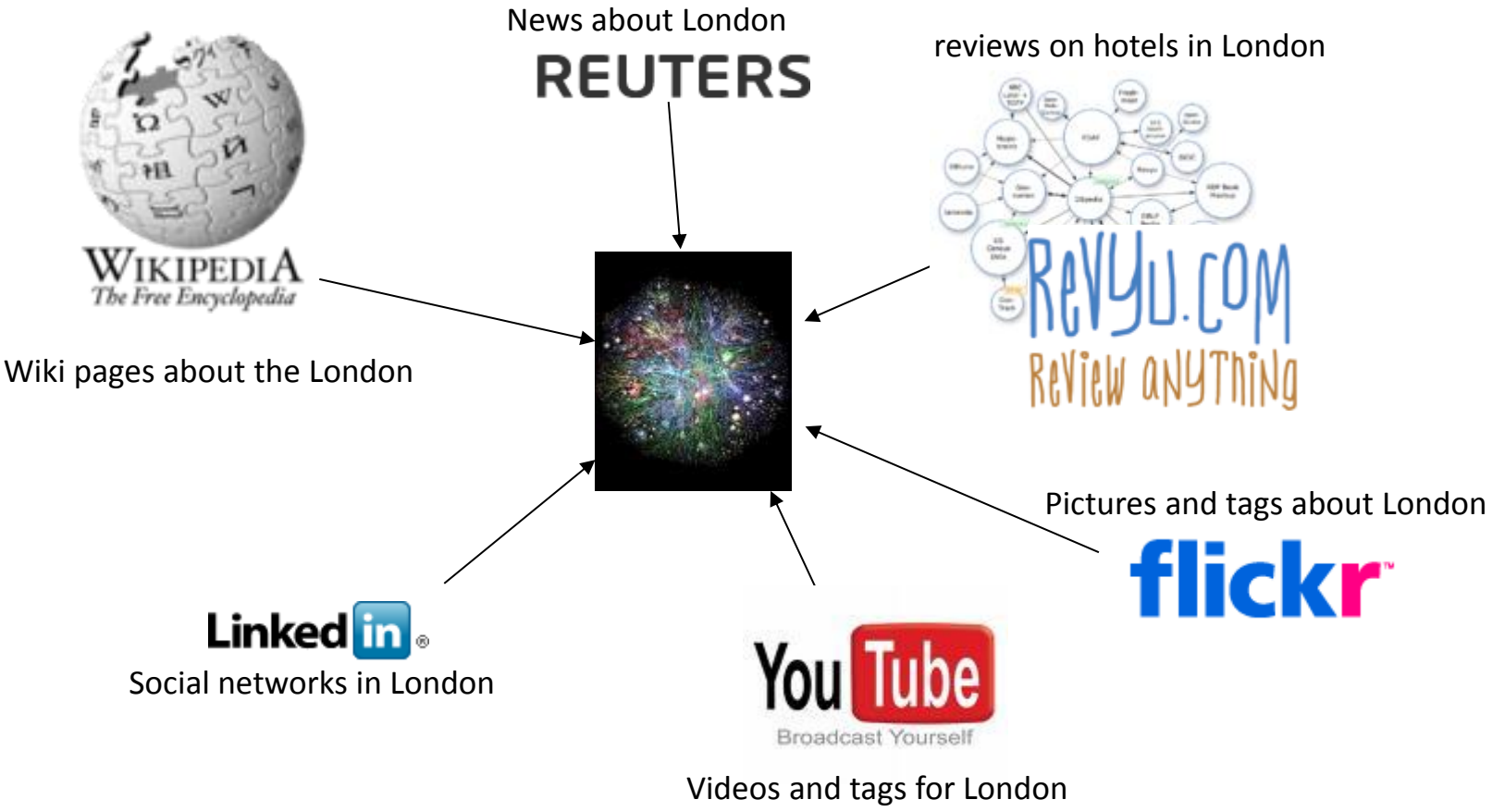
How many "entities" have the same name?

- 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
- ...

Content Providers

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



Preliminaries on Entity Resolution

Entity Resolution [Christen, TKDE 2011]:

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

Useful because:

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

Application areas:

Linked Data, Social Networks, census data,

price comparison portals

Types of Entity Resolution

The input of ER consists of entity collections that can be of two types [Christen, TKDE 2011]:

- 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

The input of ER consists of entity collections that can be of two types [Christen, TKDE 2011]:

- 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

Based on the quality of input, we distinguish ER into 3 sub-tasks:

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

} Equivalent to **Dirty ER** (a.k.a. ***Deduplication*** in databases)

Computational cost

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

ER does not scale well to large entity collections (e.g., Web Data).

Computational cost

ER is an inherently quadratic problem (i.e., $O(n^2)$):

every entity has to be compared with all others

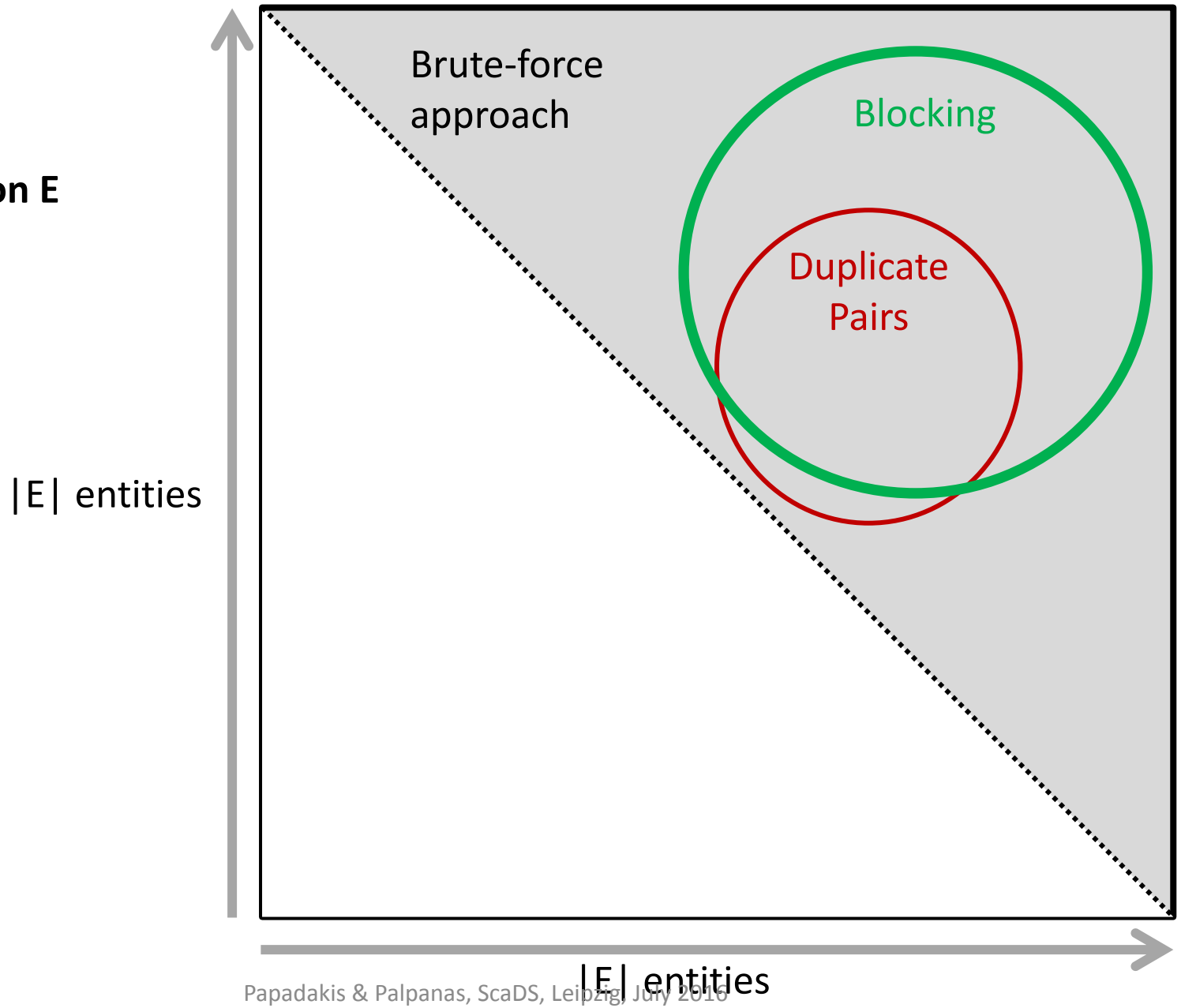
ER does not scale well to large entity collections (e.g., Web Data)

Solution: Blocking

- group similar entities into blocks
- execute comparisons only inside each block
  - complexity is now quadratic to the size of the block (much smaller than dataset size!)

Computational cost

Input:
Entity Collection E



Example of Computational cost

DBPedia 3.0rc ↔ DBPedia 3.4

1.2 million entities ↔ 2.2 million entities

Entity matching: Jaccard similarity of all tokens
Cost per comparison: 0.045 milliseconds (average of 0.1 billion comparisons)

Brute-force approach

Comparisons: $2.58 \cdot 10^{12}$
Recall: 100%
Running time: 1,344 days → **3.7 years**

Optimized Token Blocking Workflow

Overhead time: 4 hours
Comparisons: $8.95 \cdot 10^{6}$
Recall: 99%
Total Running time: **10 hours**

Example of Computational cost

DBPedia 3.0rc ↔ DBPedia 3.4

1.2 million entities ↔ 2.2 million entities

Entity matching: Jaccard similarity of all tokens
Cost per comparison: 0.045 milliseconds (average of 0.1 billion comparisons)

Brute-force approach

Comparisons: $2.58 \cdot 10^{12}$
Recall: 100%
Running time: 1,344 days → **3.7 years**

Optimized Token Blocking Workflow

Overhead time: 4 hours
Comparisons: $8.95 \cdot 10^{6}$
Recall: 99%
Total Running time: **10 hours**

Outline

1. Introduction to Blocking
2. Blocking Methods for Relational Data
3. Blocking Methods for Web Data
4. Block Processing Techniques
5. Meta-blocking
6. Challenges
7. JedAI Toolkit
8. Conclusions

Part 1:

Introduction to Blocking

Fundamental Assumptions

1. Every entity profile consists of a *uniquely identified* set of name-value pairs.

2. Every entity profile corresponds to a single real-world object.

3. Two matching profiles are *detected* as long as they co-occur in at least one block → **entity matching** is an orthogonal problem.

4. Focus on string values.

General Principles

1. Represent each entity by *one or more* blocking keys.
2. Place into blocks all entities having the *same* or *similar* blocking key.

Measures for assessing block quality [Christen, TKDE 2011]:

- Pairs Completeness: $PC = \frac{detected\ matches}{existing\ matches}$ (optimistic recall)
- Pairs Quality: $PQ = \frac{detected\ matches}{executed\ comparisons}$ (pessimistic precision)

Trade-off!

Problem Definition

Given one dirty (Dirty ER), or two clean (Clean-Clean ER) entity collections, cluster their profiles into blocks and process them so that both *Pairs Completeness* (**PC**) and *Pairs Quality* (**PQ**) are maximized.

caution:

- Emphasis on Pairs Completeness (PC).
  - if two entities are matching then they should coincide at some block

Blocking Techniques Taxonomy

1. Performance-wise
   - Exact methods
   - Approximate methods
2. Functionality-wise
   - Supervised methods
   - Unsupervised methods
3. Blocks-wise
   - Disjoint blocks
   - Overlapping blocks
     - Redundancy-neutral
     - Redundancy-positive
     - Redundancy-negative
4. Signature-wise
   - Schema-based
   - Schema-agnostic

Performance-wise Categorization

1. **Exact Blocking Methods**
   - Maximize PQ for PC = 100%
   - **Closed**-world assumption
   - E.g., for bibliographical records , **s ≡ t** if:

     JaccardSimilarity(s.title, t.title) > 0.80 AND

     EditDistance(s.venue, t.venue) < 3
   - Existing methods:
     - **Silk** → filtering technique for edit distance
     - **LIMES** → triangle inequality for similarity metrics
2. **Approximate Blocking Methods**
   - PC < 100% → high PQ
   - **Open**-world assumption

Performance-wise Categorization

1. **Exact Blocking Methods**
   - Maximize PQ for PC = 100%
   - **Closed**-world assumption
   - E.g., for bibliographical records , **s ≡ t** if:

     JaccardSimilarity(s.title, t.title) > 0.80 AND

     EditDistance(s.venue, t.venue) < 3
   - Existing methods:
     - **Silk** → filtering technique for edit distance
     - **LIMES** → triangle inequality for similarity metrics
2. **Approximate Blocking Methods**
   - PC < 100% → high PQ
   - **Open**-world assumption

our focus

Functionality-wise Categorization

1. **Supervised** Methods

- Goal: learn the best blocking keys from a training set
- Approach: identify best combination of attribute names and transformations
- E.g., CBLOCK [Sarma et. al, CIKM 2012],

  [Bilenko et. al., ICDM 2006], [Michelson et. al., AAAI 2006]
- Drawbacks:
  - labelled data
  - domain-dependent

2. **Unsupervised** Methods

- Generic, popular methods

Functionality-wise Categorization

1. **Supervised** Methods

- Goal: learn the best blocking keys from a training set
- Approach: identify best combination of attribute names and transformations
- E.g., CBLOCK [Sarma et. al, CIKM 2012],

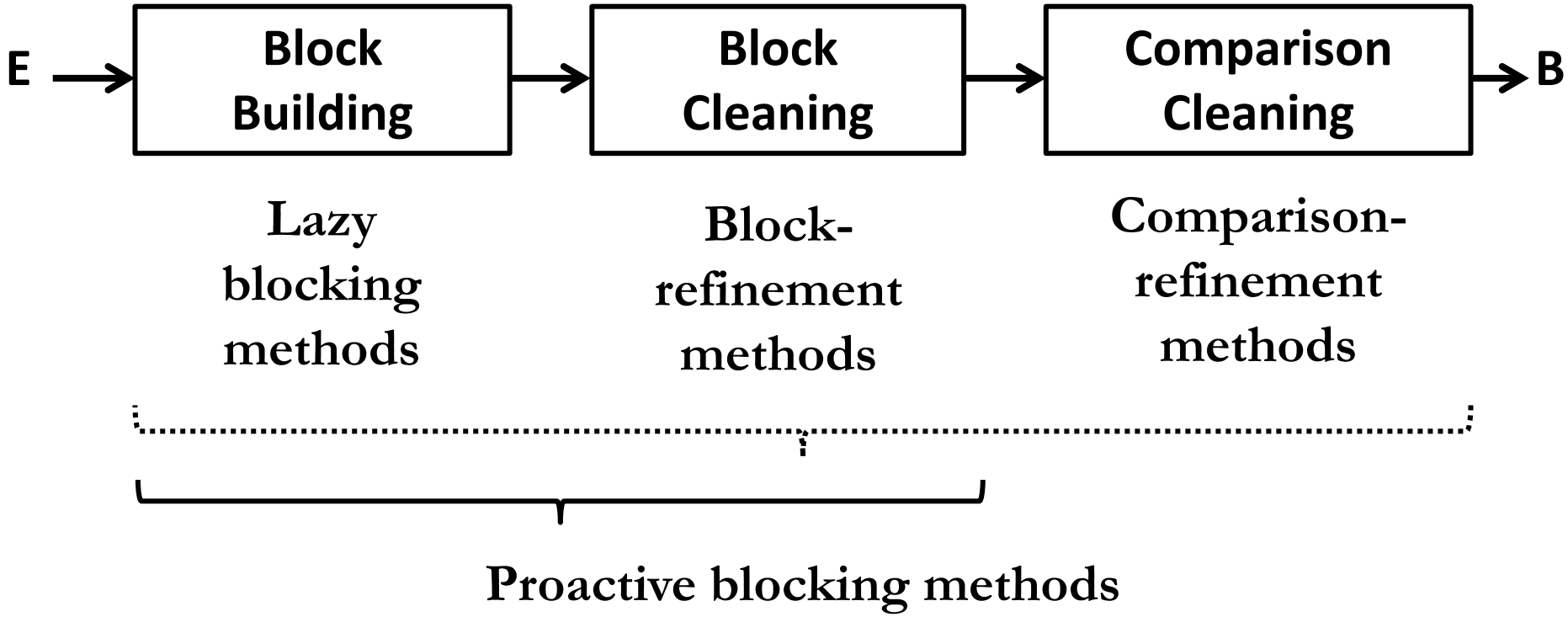
  [Bilenko et. al., ICDM 2006], [Michelson et. al., AAAI 2006]
- Drawbacks:
  - labelled data
  - domain-dependent

2. **Unsupervised** Methods

- Generic, popular methods

our focus

Blocking Workflow [Papadakis et. al., VLDB 2016]



Blocks- and Signature-wise Categorization of Block Building Methods

	Disjoint Blocks	**Overlapping Blocks**		
		Redundancy-negative	**Redundancy-neutral**	**Redundancy-positive**
Schema-based	Standard Blocking	(Extended) Canopy Clustering	1. (Extended) Sorted Neighborhood 2. MFIBlocks	1. (Extended) Q-grams Blocking 2. (Extended) Suffix Arrays
Schema-agnostic	-	-	-	1. Token Blocking 2. Agnostic Clustering 3. TYPiMatch 4. URI Semantics Blocking

Block Processing Methods

[Papadakis et. al., VLDB 2016]

Mostly for redundancy-positive block building methods.

Block Cleaning

- Block-level
  - constraints on block characteristics
- Entity-level
  - constraints on entity characteristics

Comparison Cleaning

- Redundant comparisons
  - repeated across different blocks
- Superfluous comparisons
  - Involve non-matching entities

Part 2:

Block Building for Relational Data

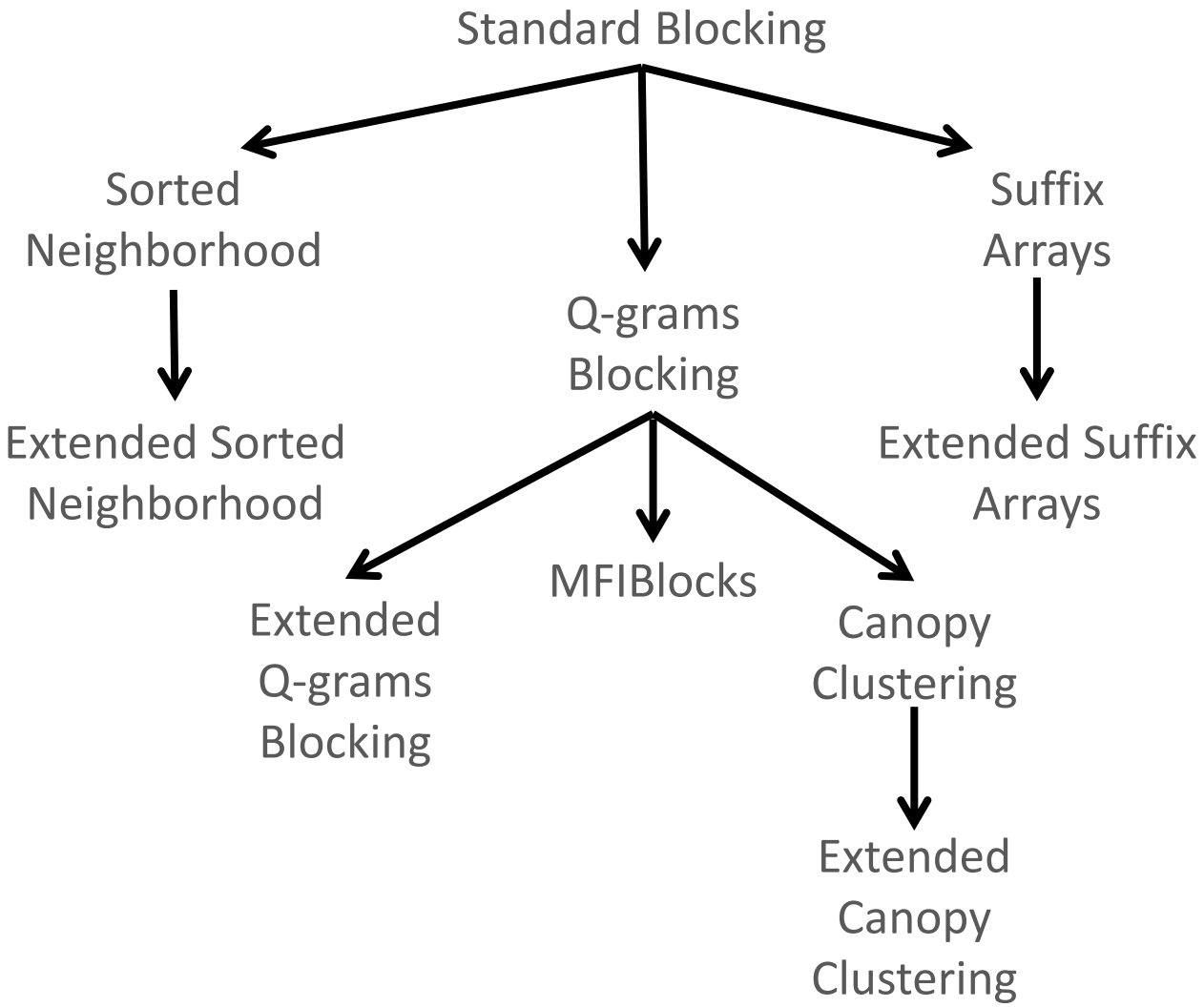
General Principles

Mostly schema-based techniques.

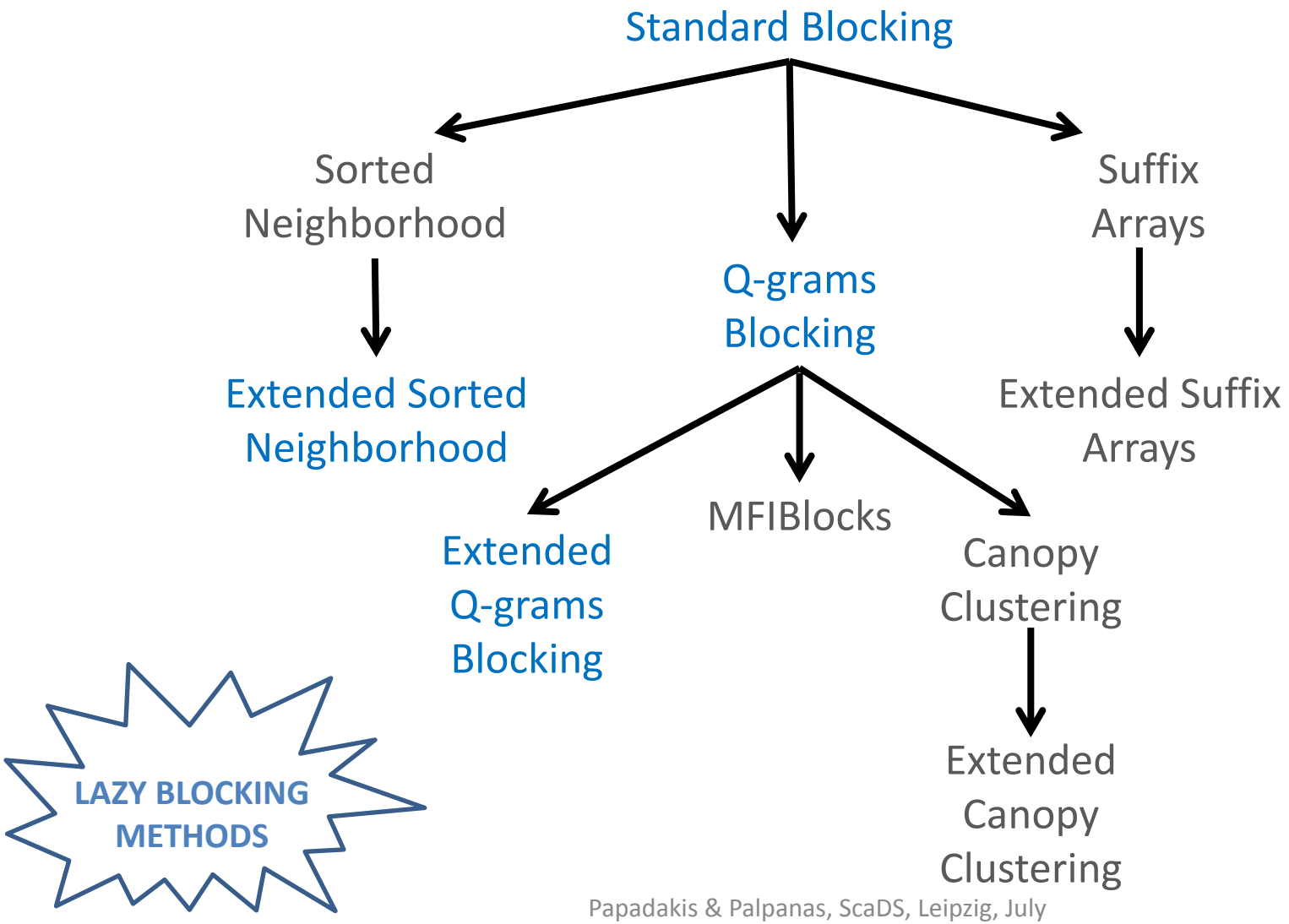
Rely on two assumptions:

1. A-priori known schema → no noise in attribute names.
2. For each attribute name we know some metadata:
   - level of noise (e.g., spelling mistakes, false or missing values)
   - distinctiveness of values

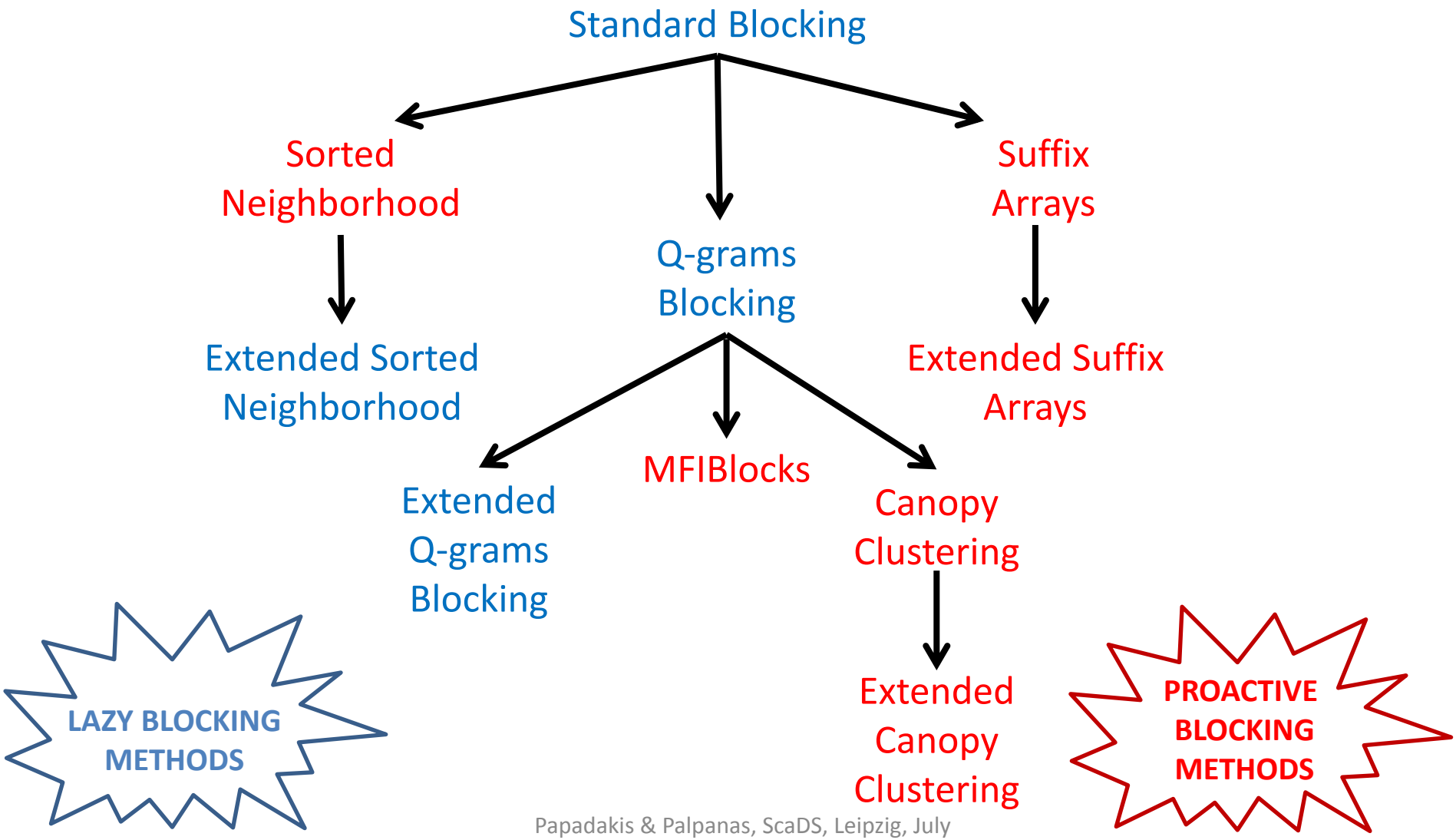
Overview of Schema-based Methods



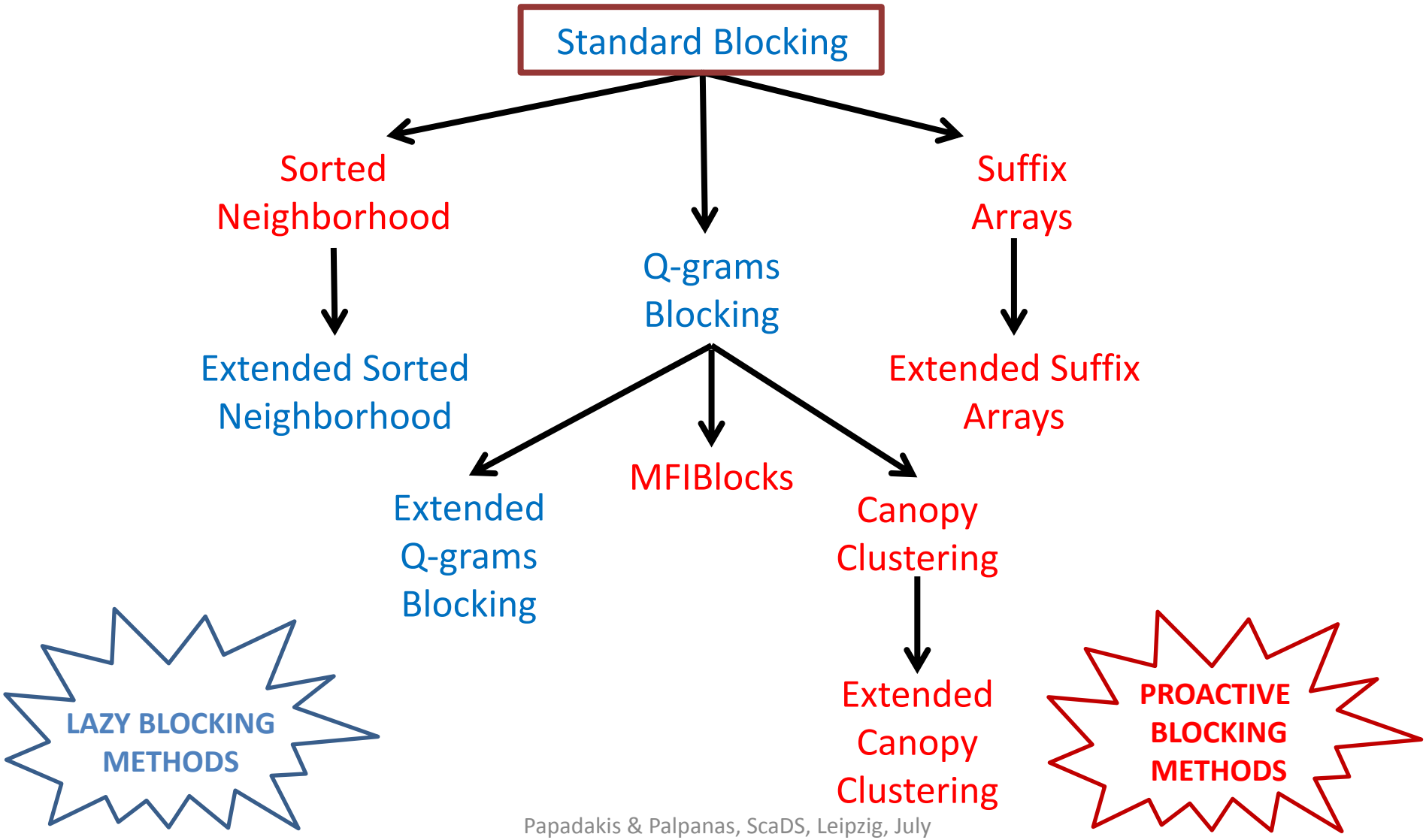
Overview of Schema-based Methods



Overview of Schema-based Methods



Overview of Schema-based Methods



Standard Blocking [Fellegi et. al., JASS 1969]

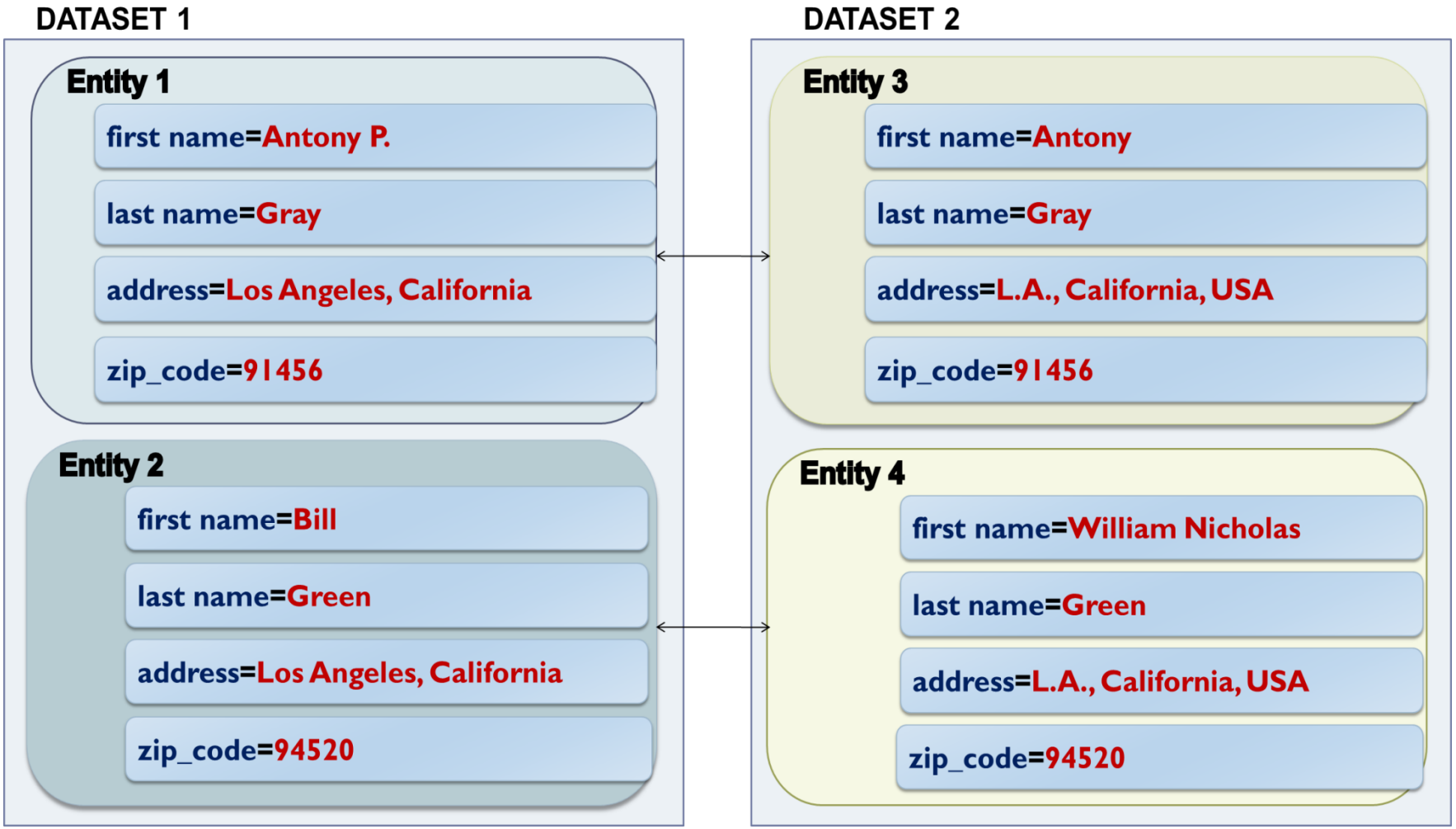
Earliest, simplest form of blocking.

Algorithm:

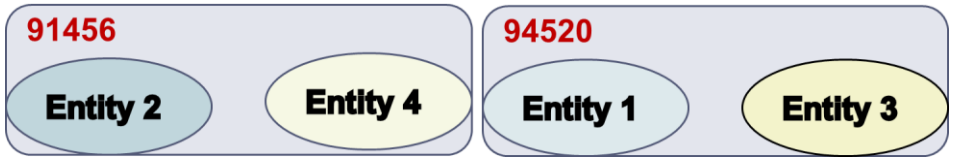
1. Select the most appropriate attribute name(s) w.r.t. noise and distinctiveness.
2. Transform the corresponding value(s) into a Blocking Key (BK)
3. For each BK, create one block that contains all entities having this BK in their transformation.

Works as a hash function! → Blocks on the **equality** of BKs

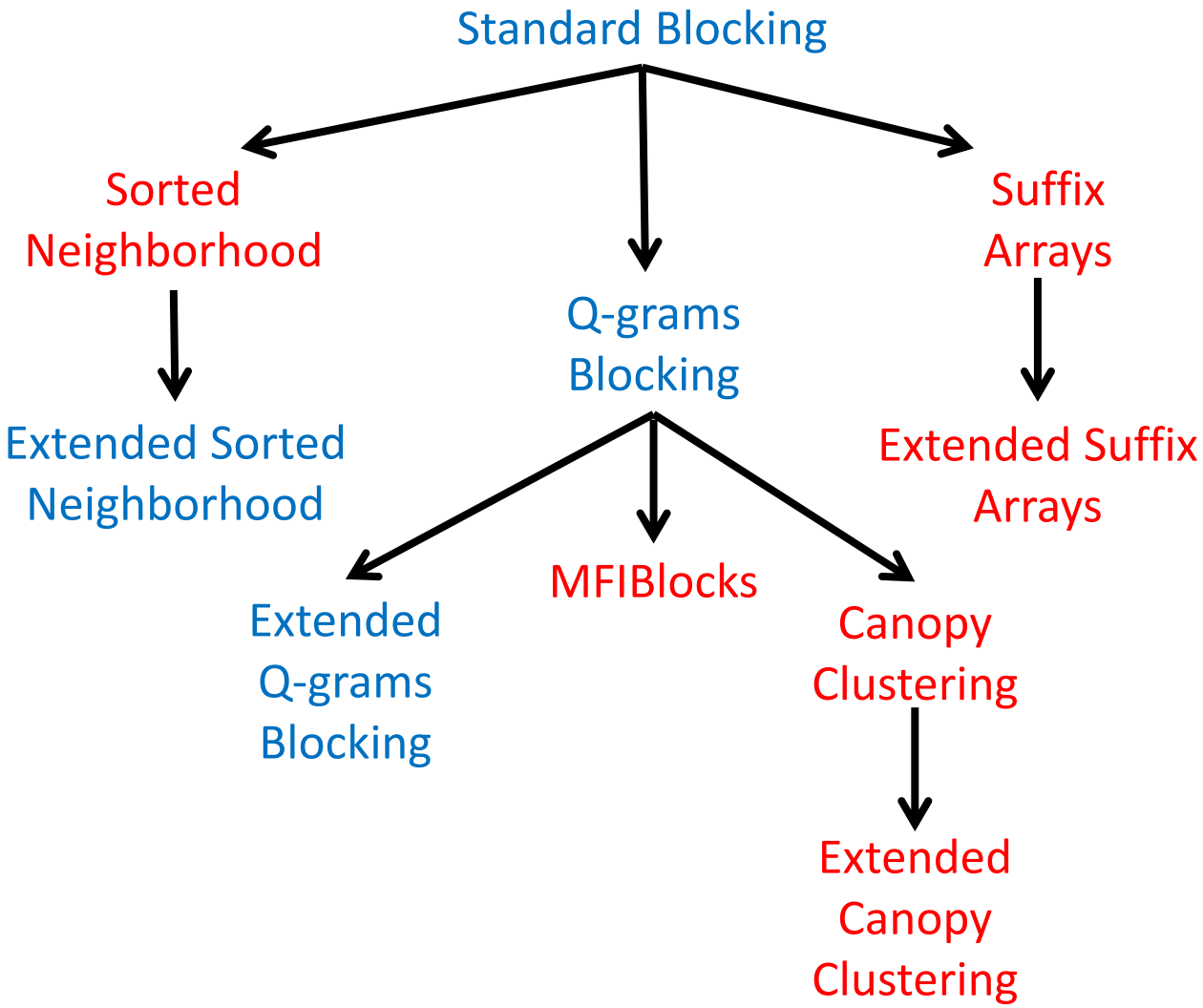
Example of Standard Blocking



Blocks on zip_code:

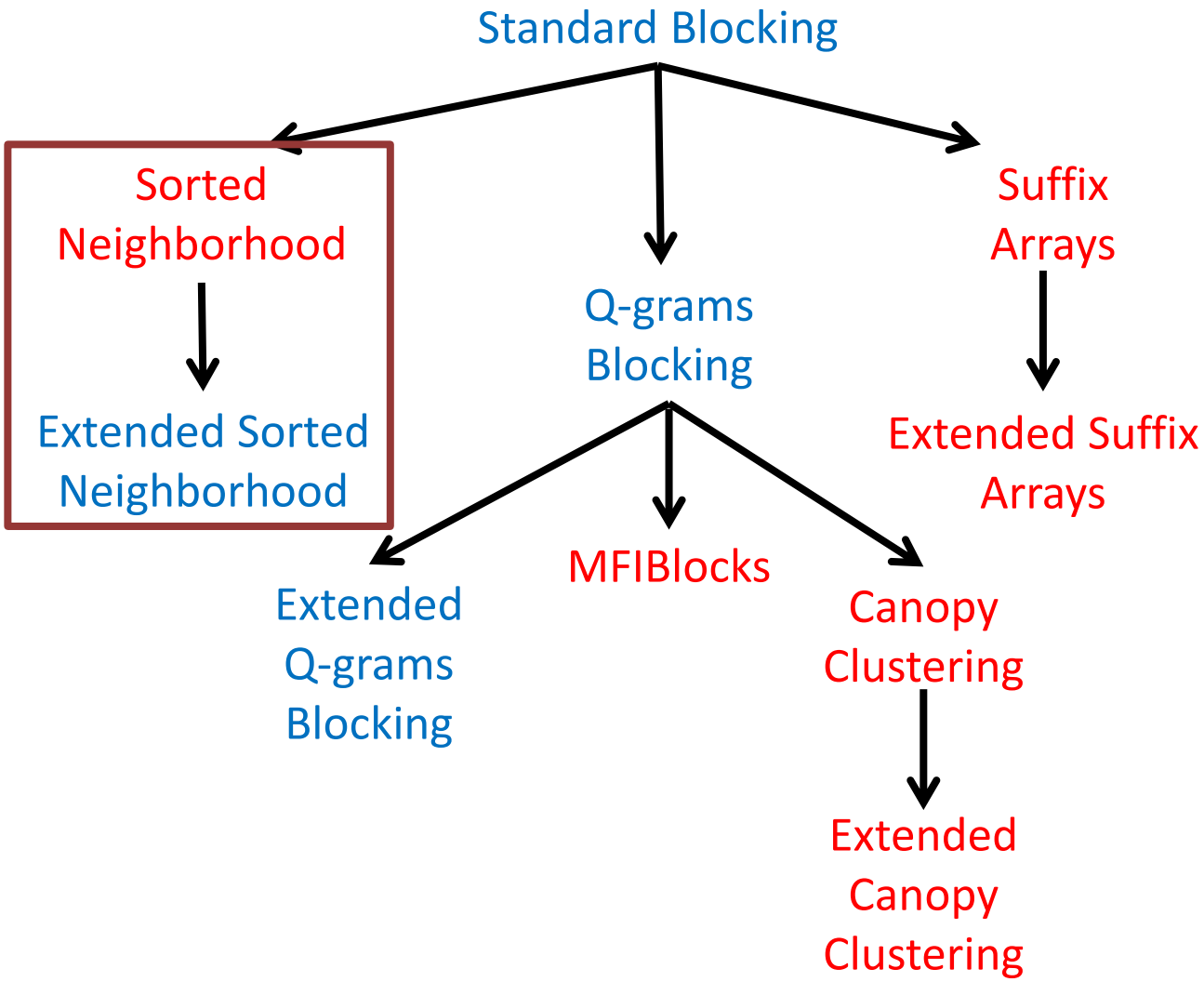


Overview of Schema-based Methods



Overview of Schema-based Methods

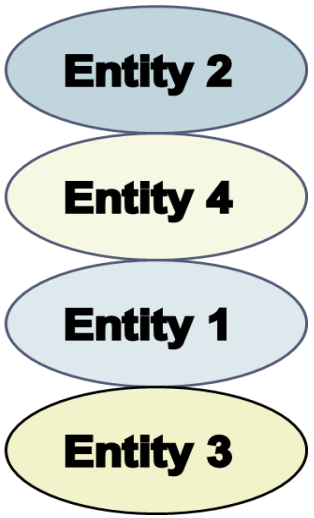
blocks contain entities with **similar** blocking keys



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

Blocks on the similarity of BKs.

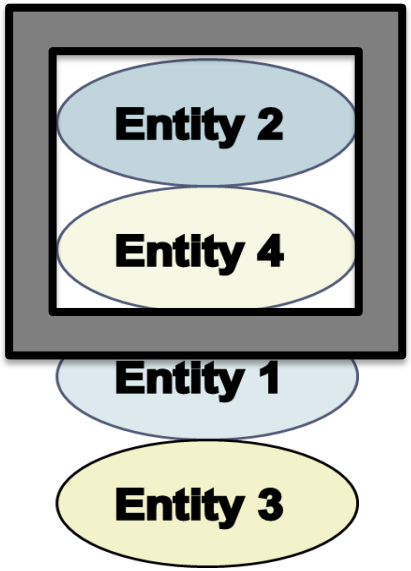
1. Entities are sorted in alphabetic order of BKs.
2. A window of fixed size slides over the sorted list of entities.
3. At each iteration, it compares the entities that co-occur within the window.



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

Blocks on the similarity of BKs.

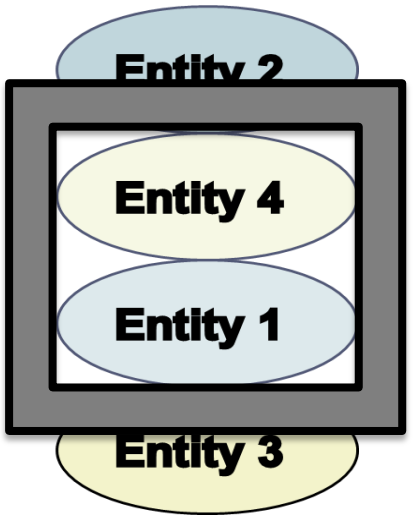
1. Entities are sorted in alphabetic order of BKs.
2. A window of fixed size slides over the sorted list of entities.
3. At each iteration, it compares the entities that co-occur within the window.



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

Blocks on the similarity of BKs.

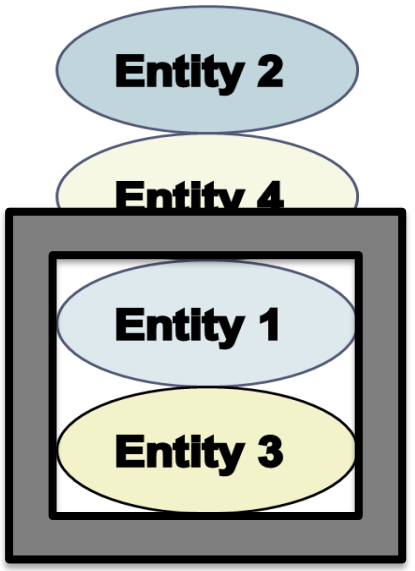
1. Entities are sorted in alphabetic order of BKs.
2. A window of fixed size slides over the sorted list of entities.
3. At each iteration, it compares the entities that co-occur within the window.



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

Blocks on the similarity of BKs.

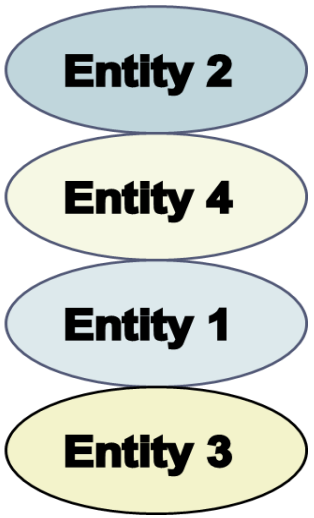
1. Entities are sorted in alphabetic order of BKs.
2. A window of fixed size slides over the sorted list of entities.
3. At each iteration, it compares the entities that co-occur within the window.



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

Blocks on the similarity of BKs.

1. Entities are sorted in alphabetic order of BKs.
2. A window of fixed size slides over the sorted list of entities.
3. At each iteration, it compares the entities that co-occur within the window.



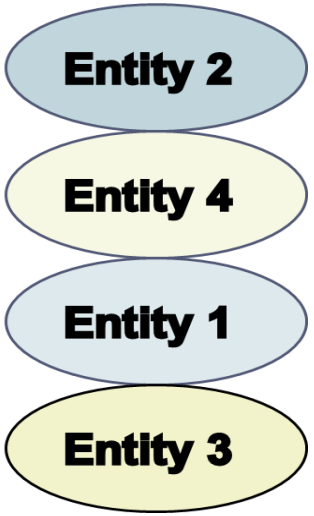
Extended Sorted Neighborhood [Christen, TKDE 2011]

2'. A window of fixed size slides over the sorted list of **BKs**.

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

Blocks on the similarity of BKs.

1. Entities are sorted in alphabetic order of BKs.
2. A window of fixed size slides over the sorted list of **entities**.
3. At each iteration, it compares the entities that co-occur within the window.



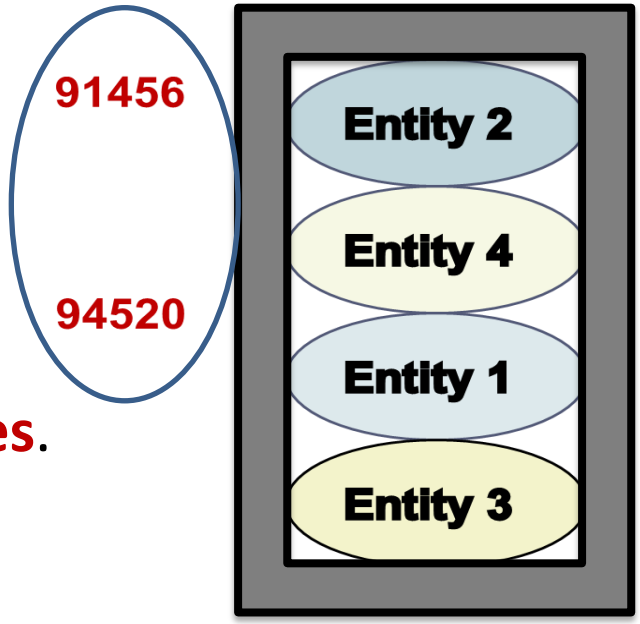
Extended Sorted Neighborhood [Christen, TKDE 2011]

2'. A window of fixed size slides over the sorted list of **BKs**.

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

Blocks on the similarity of BKs.

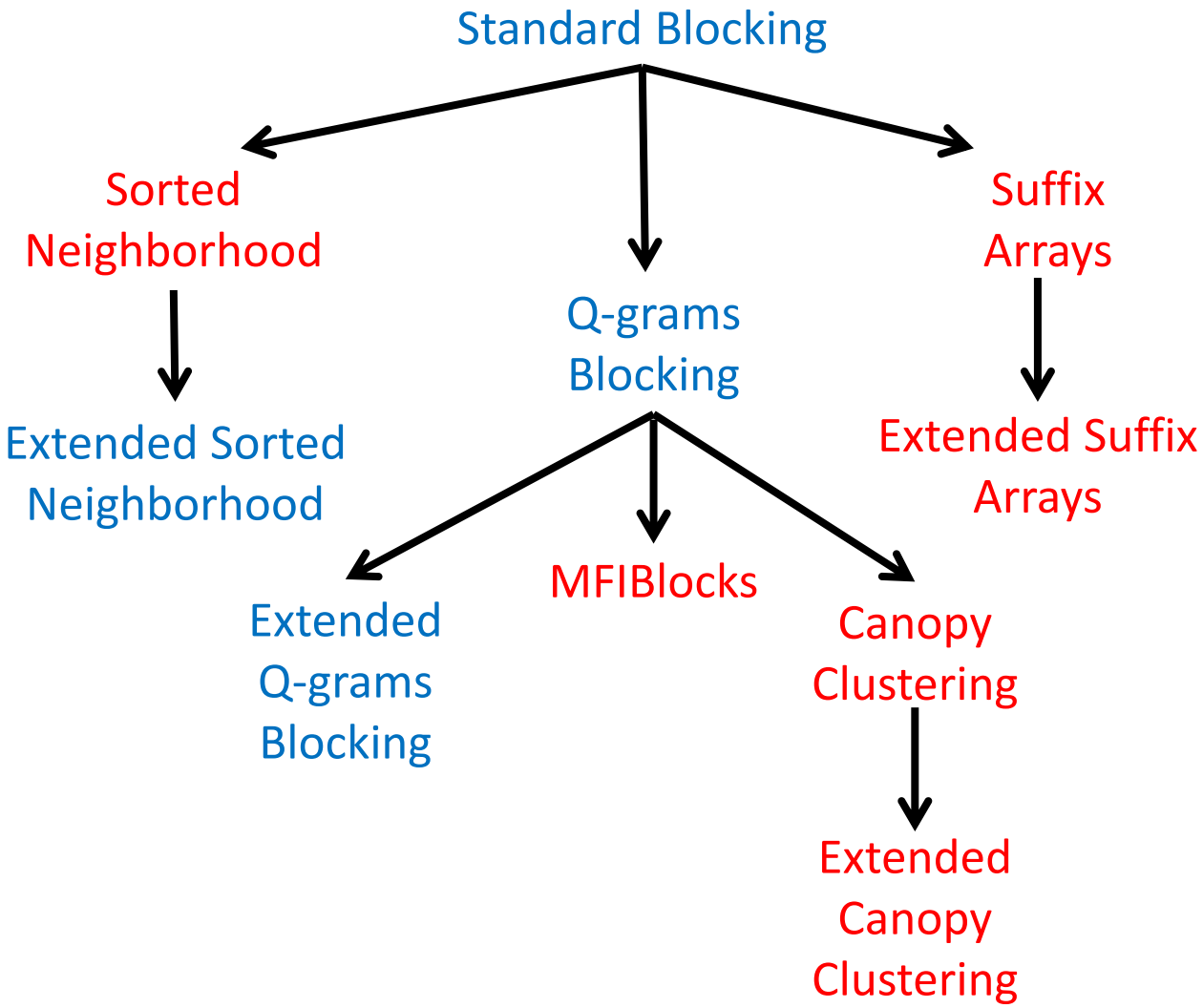
1. Entities are sorted in alphabetic order of BKs.
2. A window of fixed size slides over the sorted list of **entities**.
3. At each iteration, it compares the entities that co-occur within the window.



Extended Sorted Neighborhood [Christen, TKDE 2011]

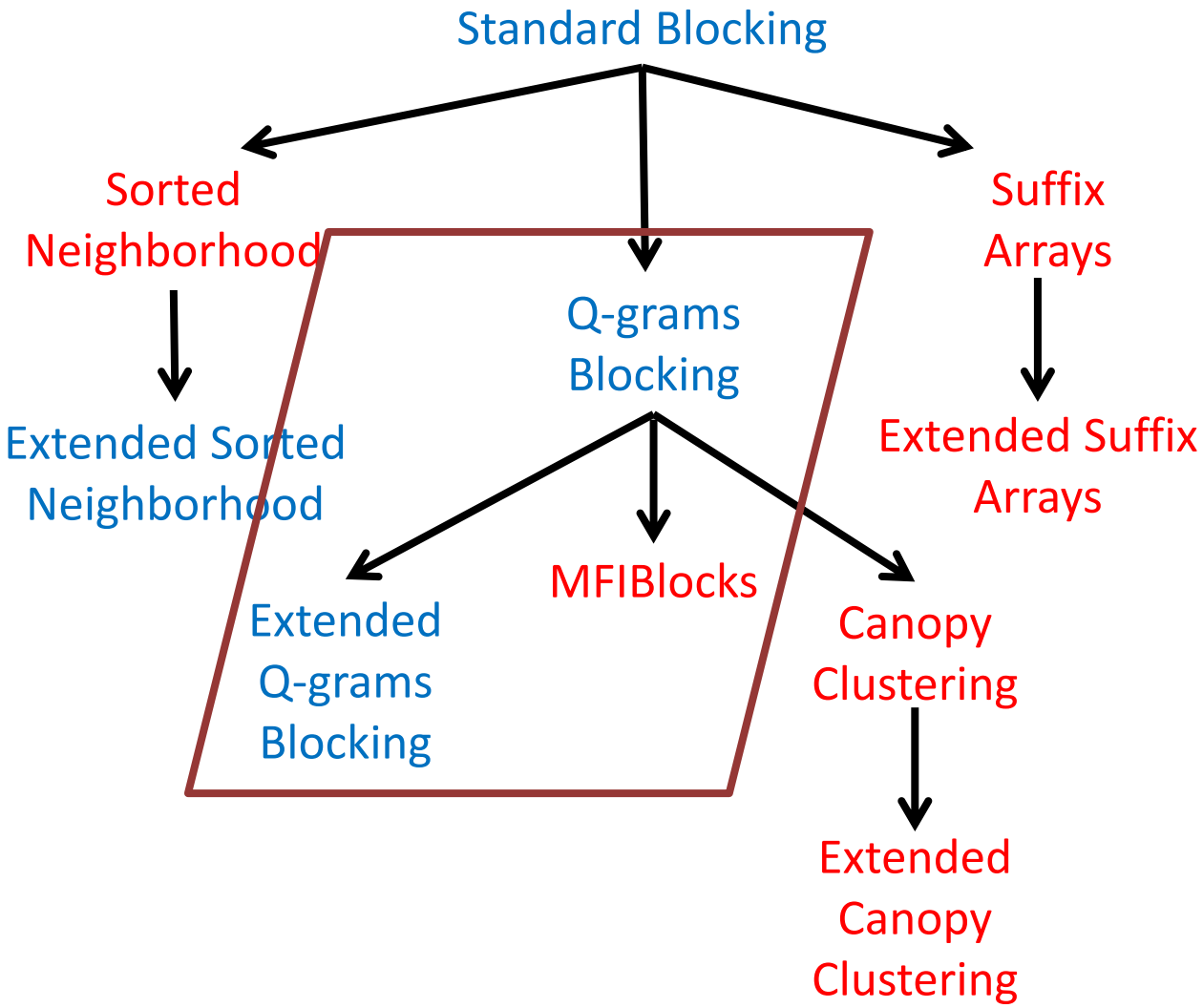
2'. A window of fixed size slides over the sorted list of **BKs**.

Overview of Schema-based Methods



Overview of Schema-based Methods

blocks contain entities with **same, or similar** blocking keys

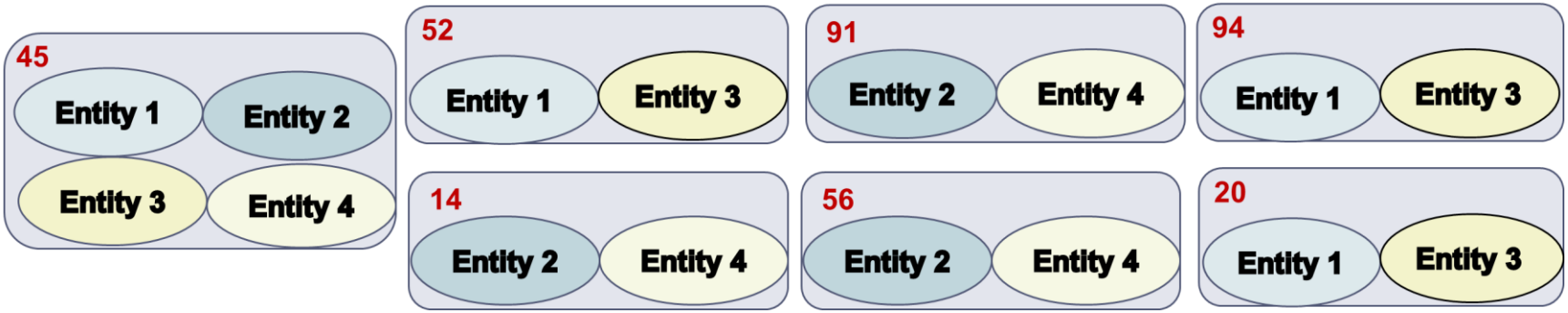


Q-grams Blocking [Gravano et. al., VLDB 2001]

Blocks on equality of BKs.

Converts every BK into the list of its *q*-grams.

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



- Advantage:

  robust to noisy BKVs
- Drawback:

  larger blocks → higher computational cost

Extended Q-grams Blocking [Baxter et. al., KDD 2003]

BKs of higher discriminativeness:

instead of individual *q*-grams, BKs from combinations of *q*-grams.

Additional parameter:

threshold $t \in (0,1)$ specifies the minimum number of *q*-grams per BK as follows: $l_{min} = max(1, \lfloor k \cdot t \rfloor)$, where k is the number of *q*-grams from the original BK

Example:

for BK= 91456, q=2 and t=0.9,

we have l_{min}=3 and the following valid BKs:

91_14_45_56

91_14_45

91_14_56

91_45_56

14_45_56

MFIBlocks [Kenig et. al., IS 2013]

Based on mining Maximum Frequent Itemsets.

Algorithm:

- Place all entities in a pool
- while (minimum_support > 2)
  - For each itemset that satisfies minimum_support
    - Create a block **b**
    - If **b** satisfies certain constraints (Block Cleaning)
      - remove its entities from the pool
      - retain the best comparisons (Comparison Cleaning)
  - decrease minimum_support

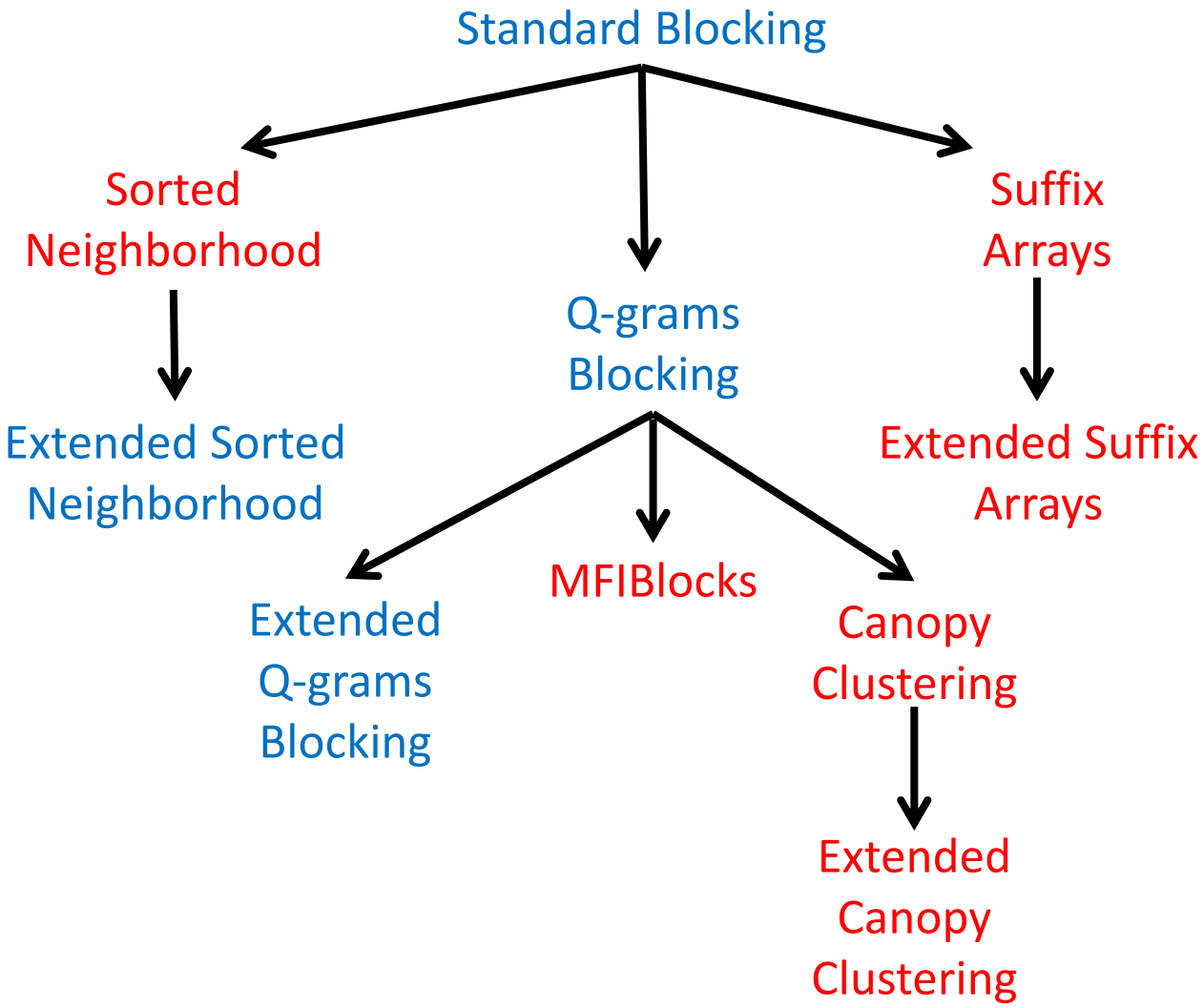
Pros:

- Usually the most effective blocking method for relational data → maximizes PQ (precision)

Cons:

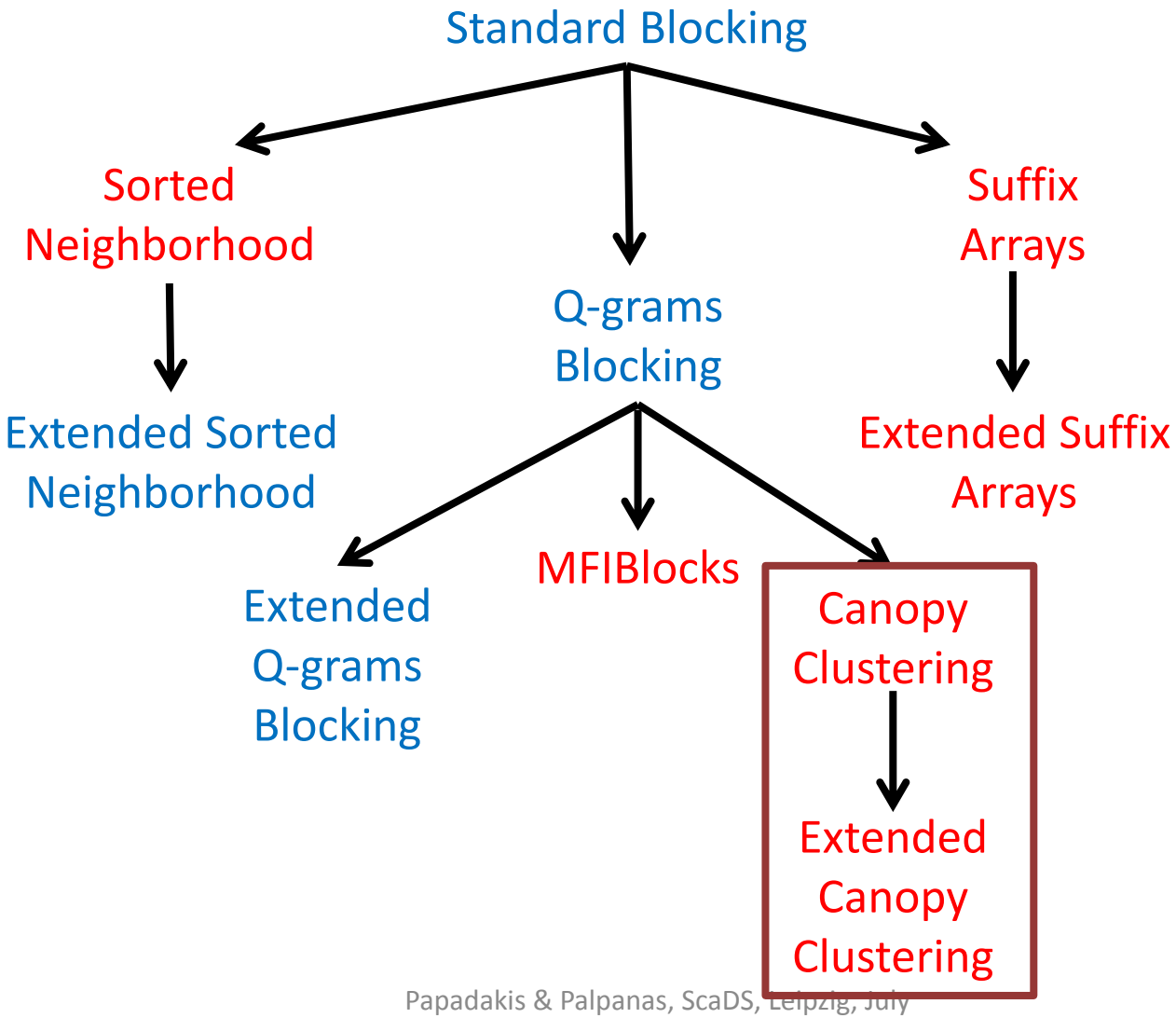
- Difficult to configure
- Time consuming

Overview of Schema-based Methods



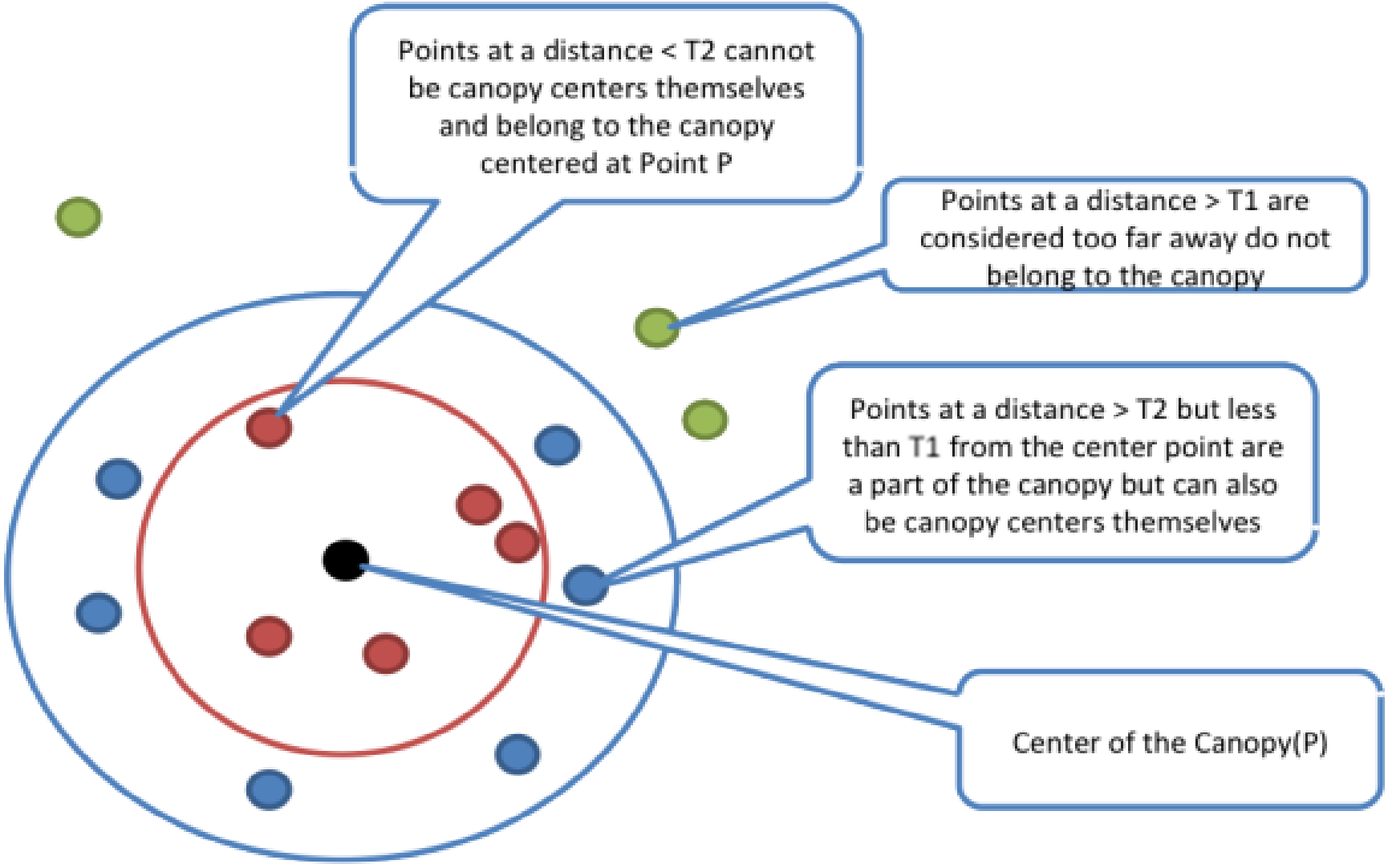
Overview of Schema-based Methods

blocks contain entities with **similar** blocking keys



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

Blocks on similarity of BKs.



Extended Canopy Clustering [Christen, TKDE 2011]

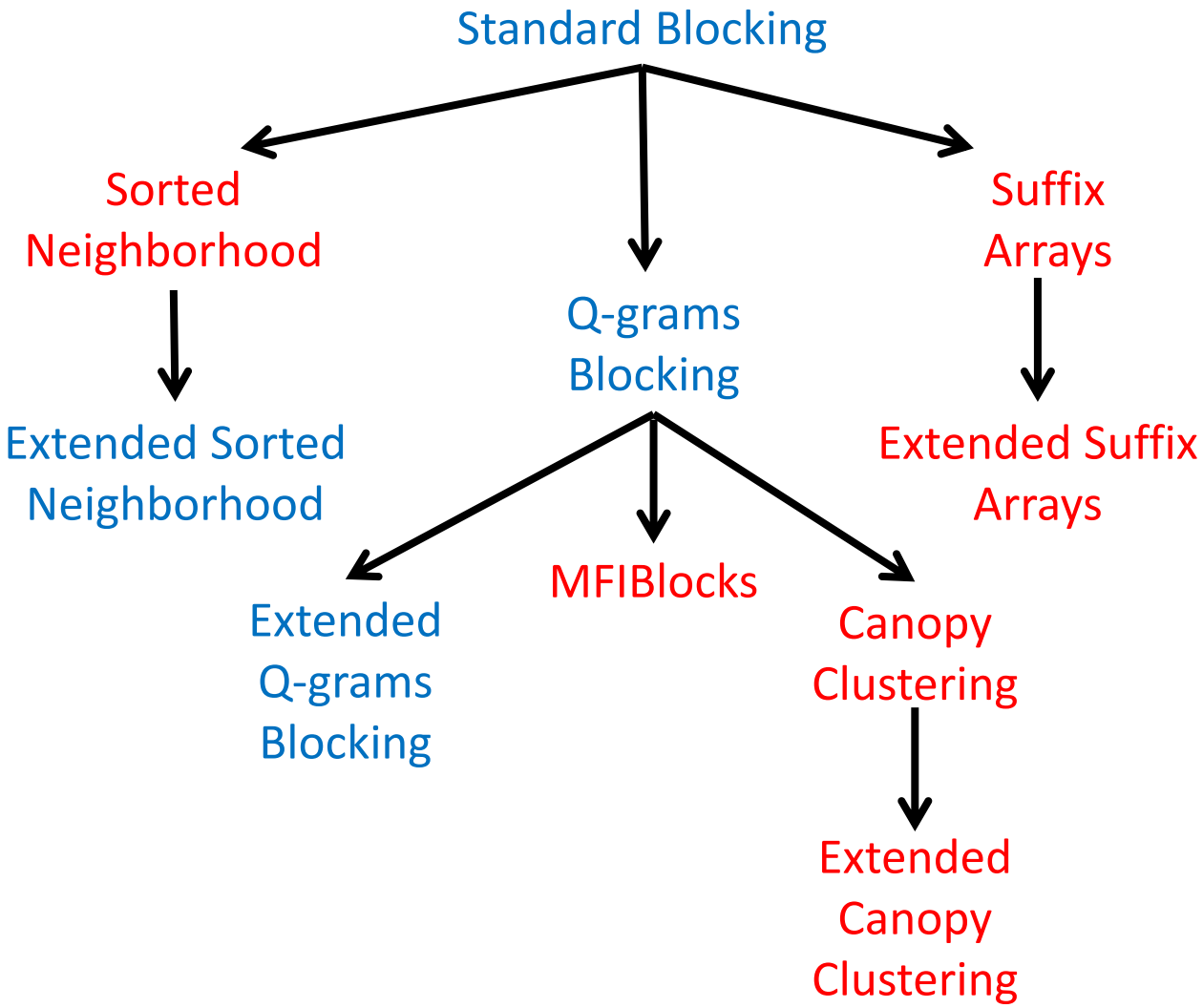
Canopy Clustering is too sensitive w.r.t. its weight thresholds:

- high values may leave many entities out of blocks.

Solution: **Extended Canopy Clustering** [Christen, TKDE 2011]

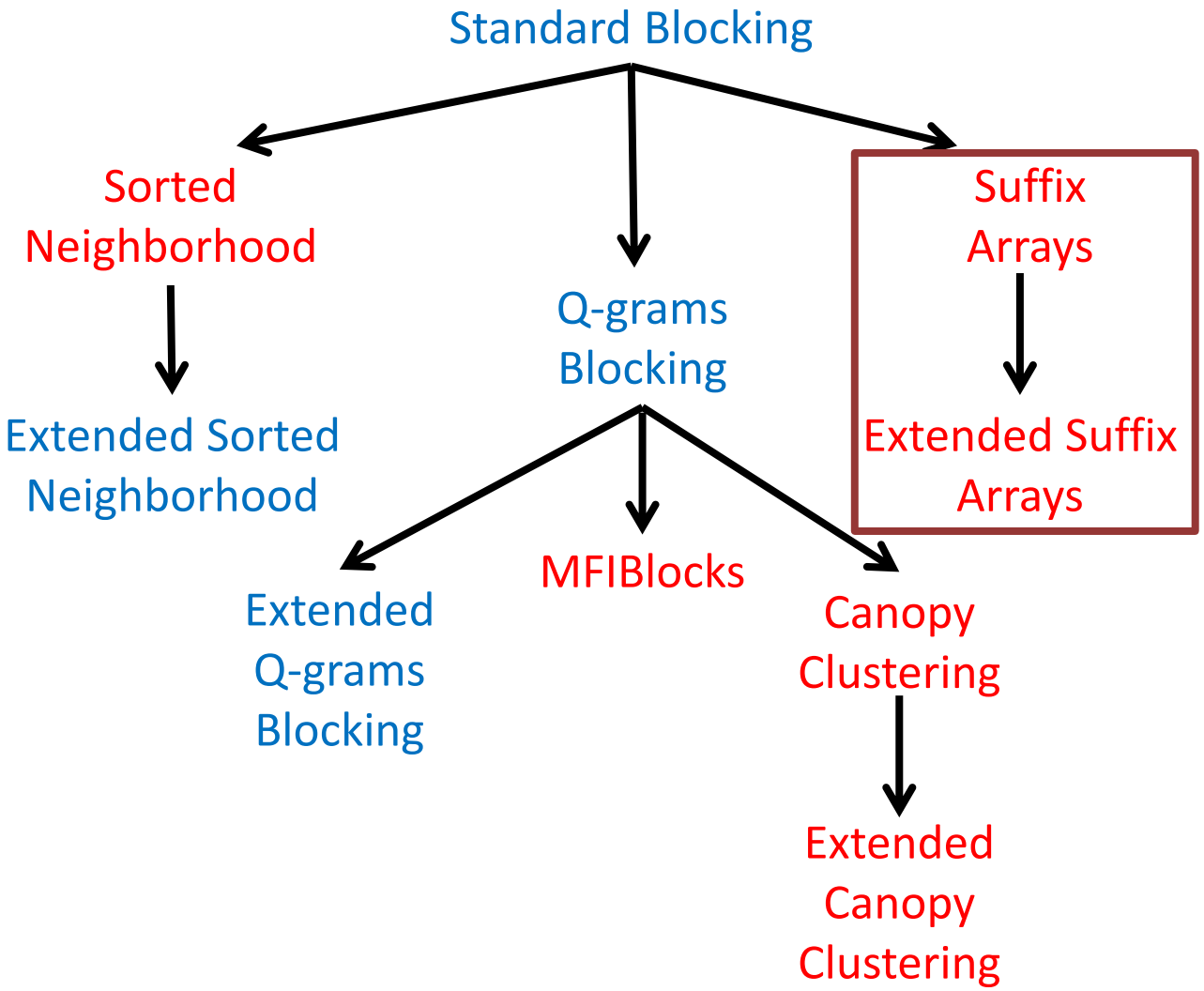
- cardinality thresholds instead of weight thresholds
- for each center of a canopy:
  - the n_1 nearest entities are placed in its block
  - the n_2 ($\leq n_1$) nearest entities are removed from the pool

Overview of Schema-based Methods



Overview of Schema-based Methods

blocks contain entities with **same** blocking keys

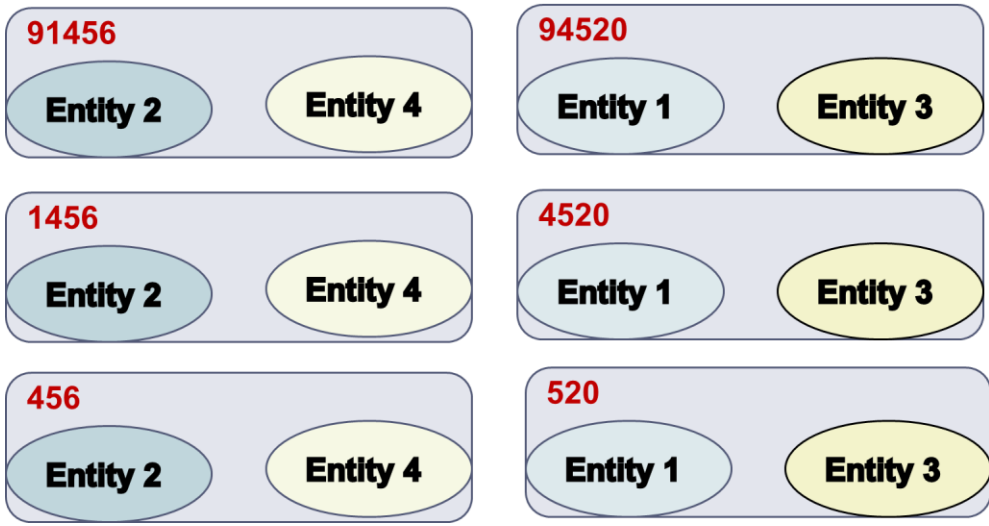


Suffix Arrays Blocking [Aizawa et. al., WIRI 2005]

Blocks on the equality of BKs.

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

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



Frequent suffixes are discarded with the help of the parameter b_M:

- specifies the maximum number of entities per block

Extended Suffix Arrays Blocking [Christen, TKDE 2011]

Goal:

support errors at the end of BKs

Solution:

consider *all substrings* (not only suffixes) with more than l_{min} characters.

For l_{min}=3, the keys *91456* and *94520* are converted to the BKs:

91456,	94520
9145,	9452
1456,	4520
914,	945
145,	452
456	520

Summary of Blocking for Databases [Christen, TKDE2011]

1. They typically employ **redundancy** to ensure higher recall in the context of noise at the cost of lower precision (more comparisons). Still, recall remains low for many datasets.

2. Several parameters to be configured

   E.g., Canopy Clustering has the following parameters:

   I. String matching method

   II. Threshold t_1

   III. Threshold t_2

3. Schema-dependent → manual definition of BKs

Improving Blocking for Databases [Papadakis et. al., VLDB 2015]

Schema-agnostic blocking keys

- Use every token as a key
- Applies to all schema-based blocking methods
- Simplifies configuration, unsupervised approach

Performance evaluation

- For lazy blocking methods →
  very high, robust recall at the cost of more comparisons
- For proactive blocking methods →
  relative recall gets higher with more comparisons,
  absolute recall depends on block constraints

Part 3:

Block Building for Web Data

Characteristics of Web Data

Voluminous, (semi-)structured datasets.

- DBPedia 2014: 3 billion triples and 38 million entities
- BTC09: 1.15 billion triples, 182 million entities.

Users are free to add attribute values and/or attribute names

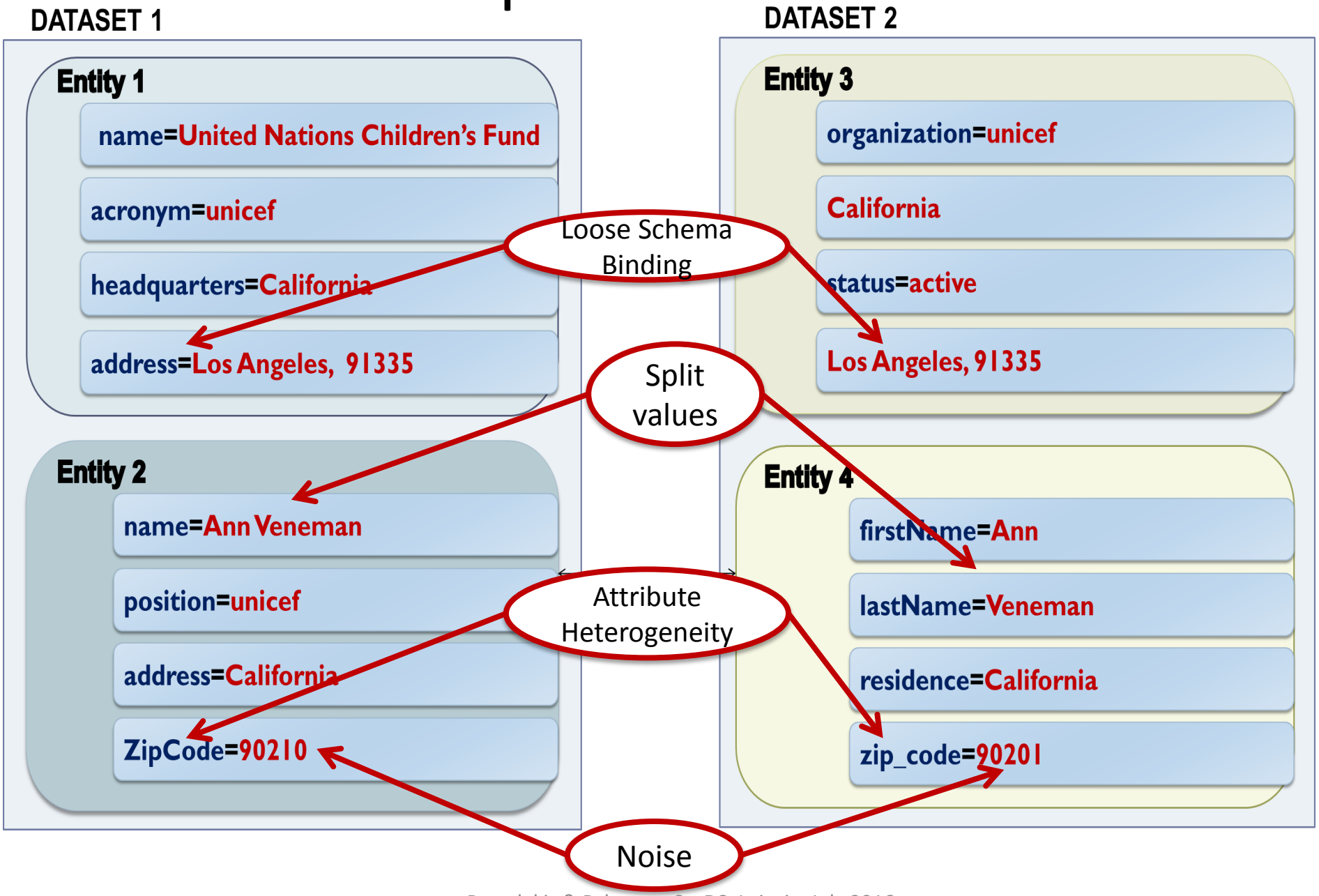
→ unprecedented levels of schema heterogeneity.

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

Several datasets produced by automatic information extraction techniques

→ noise, tag-style values.

Example of Web Data



Token Blocking [Papadakis et al., WSDM2011]

Functionality:

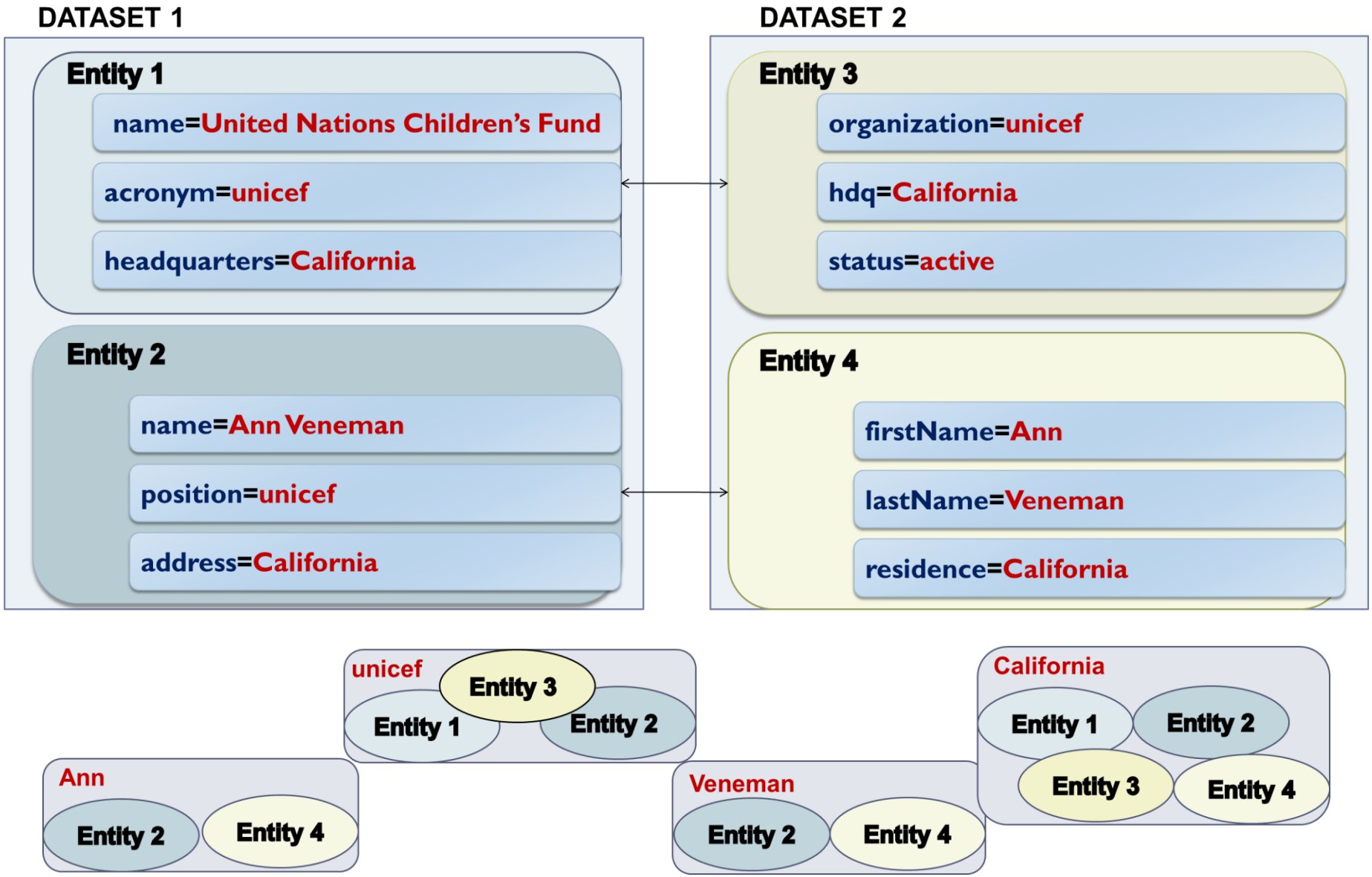
1. given an entity profile, extract all tokens that are contained in its attribute values.
2. create one block for every distinct token → each block contains all entities with the corresponding token*.

Attribute-agnostic functionality:

- completely ignores all attribute names, but considers all attribute values
- efficient implementation with the help of inverted indices
- ***parameter-free***!

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

Token Blocking Example

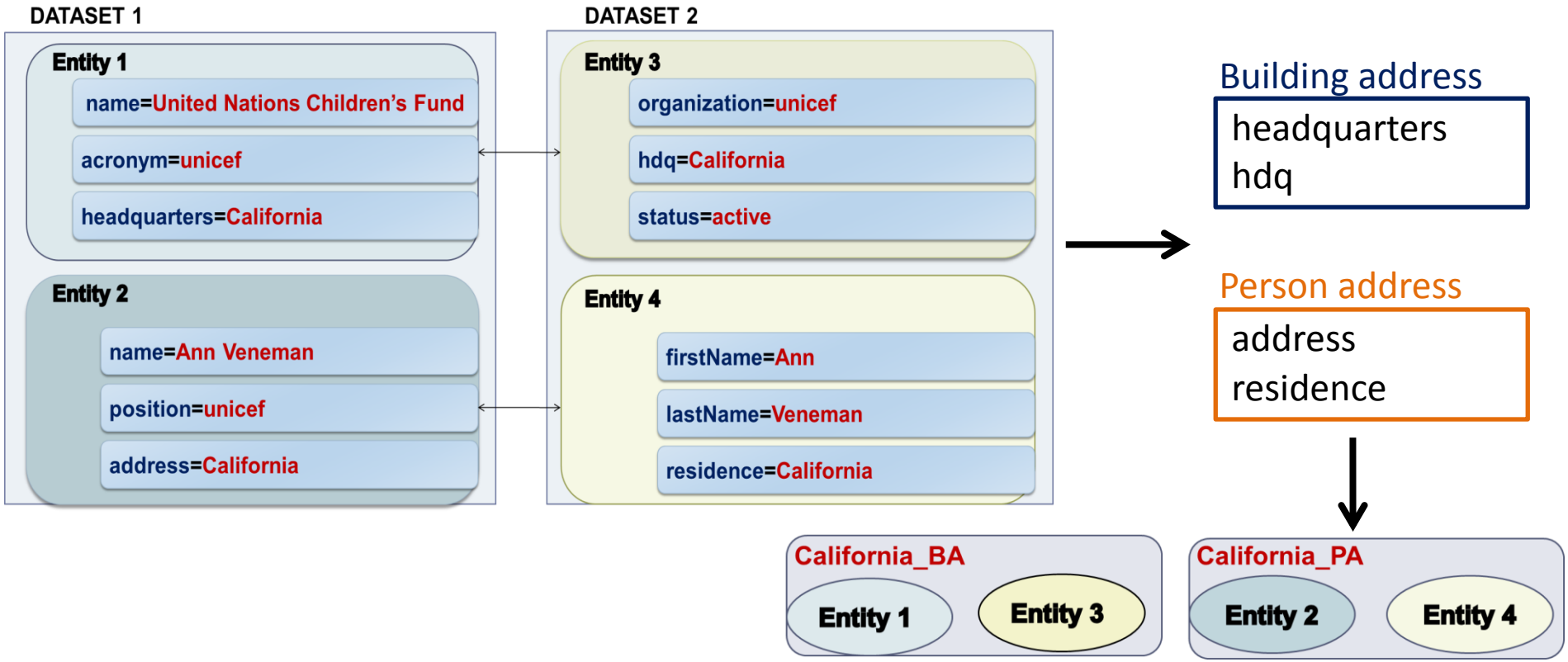


Attribute-Clustering Blocking

[Papadakis et. al., TKDE 2013]

Goal:

group attribute names into clusters s.t. we can apply Token Blocking independently inside each cluster, without affecting effectiveness
→ smaller blocks, higher efficiency.



Attribute-Clustering Blocking

Algorithm

- Create a graph, where every node represents an attribute name and its attribute values
- For each attribute name/node n_i
  - Find the most similar node n_j
  - If $sim(n_i,n_j) > 0$, add an edge $<n_i,n_j>$
- Extract connected components
- Put all singleton nodes in a “glue” cluster

Parameters

1. Representation model
   - Character n-grams, Character n-gram graphs, Tokens
2. Similarity Metric
   - Jaccard, Graph Value Similarity, TF-IDF

Attribute-Clustering vs Schema Matching

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

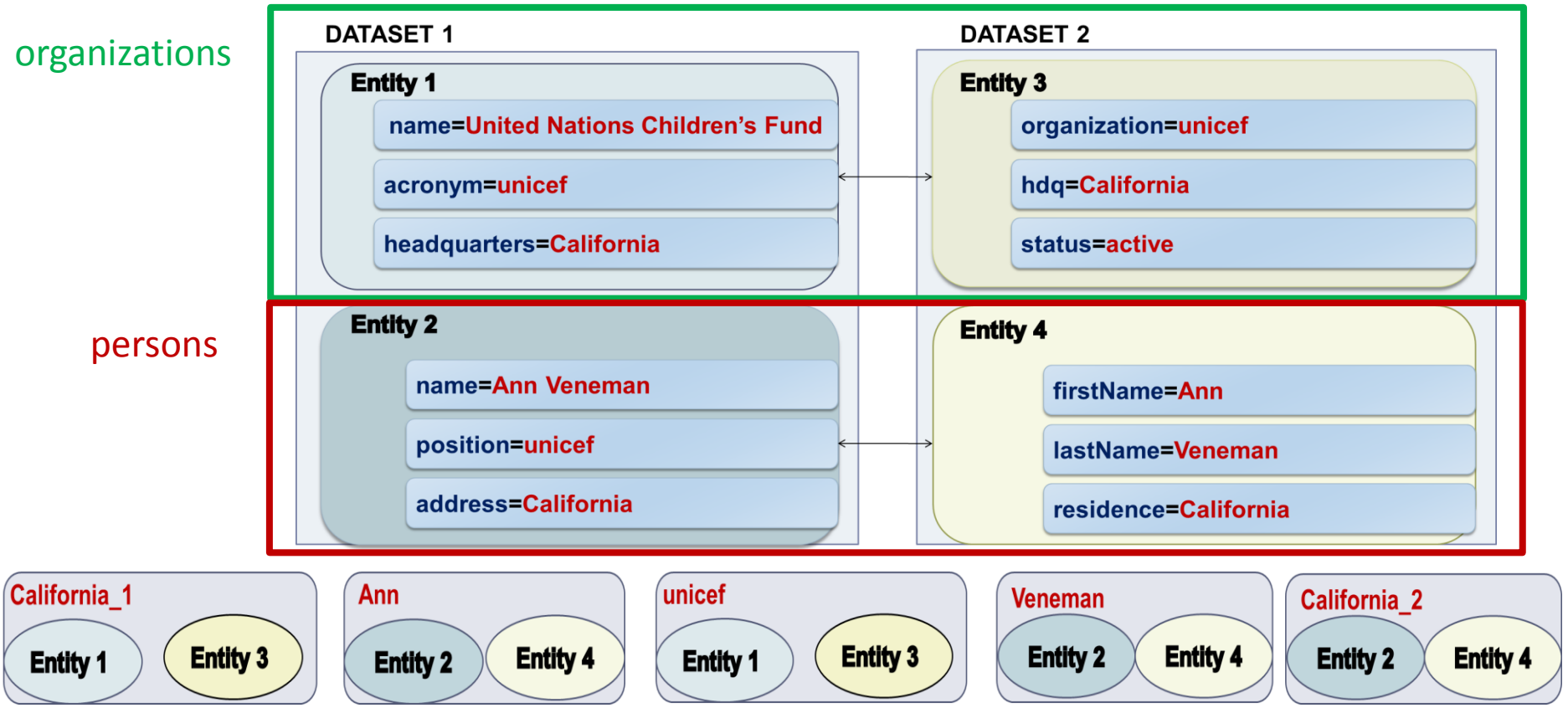
2. All singleton attribute names are associated with each other

3. Unlike Schema Matching, it scales to the very high levels of heterogeneity of Web Data
   - because of the above simplifying assumptions

TYPiMatch [Ma et. al., WSDM 2013]

Goal:

cluster entities into *overlapping types* and apply Token Blocking to the values of the best attribute for each type.



TYPiMatch

Algorithm:

1. Create a directed graph ***G***, where nodes correspond to tokens, and edges connect those co-occurring in the same entity profile, weighted according to conditional co-occurrence probability.
2. Convert **G** to undirected graph **G'** and get maximal cliques (parameter ϑ).
3. Create an undirected graph **G''**, where nodes correspond to cliques and edges connect the frequently co-occurring cliques (parameter ε).
4. Get connected components to form entity types.
5. Get best attribute name for each type using an entropy-based criterion.

Evidence for Semantic Web Blocking

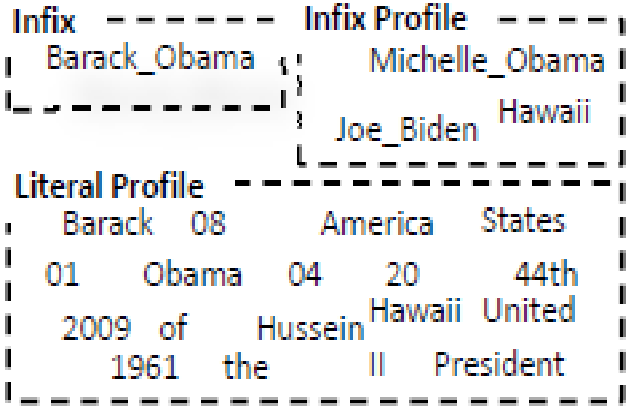
For Semantic Web data, three sources of evidence create blocks of lower redundancy than Token Blocking:

1. Infix

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. Infix Profile
3. Literal Profile

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>



Algorithm for URI decomposition in PI(S)-form in [Papadakis et al., iiWAS 2010].

URI Semantics Blocking [Papadakis et al., WSDM2012]

The above sources of evidence lead to 3 parameter-free 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 atomic methods have limited coverage and, thus, low effectiveness (e.g., Infix Blocking does not cover blank nodes).

However, they are complementary and can be combined into composite blocking methods with high robustness and effectiveness!

Summary of Blocking for Web Data

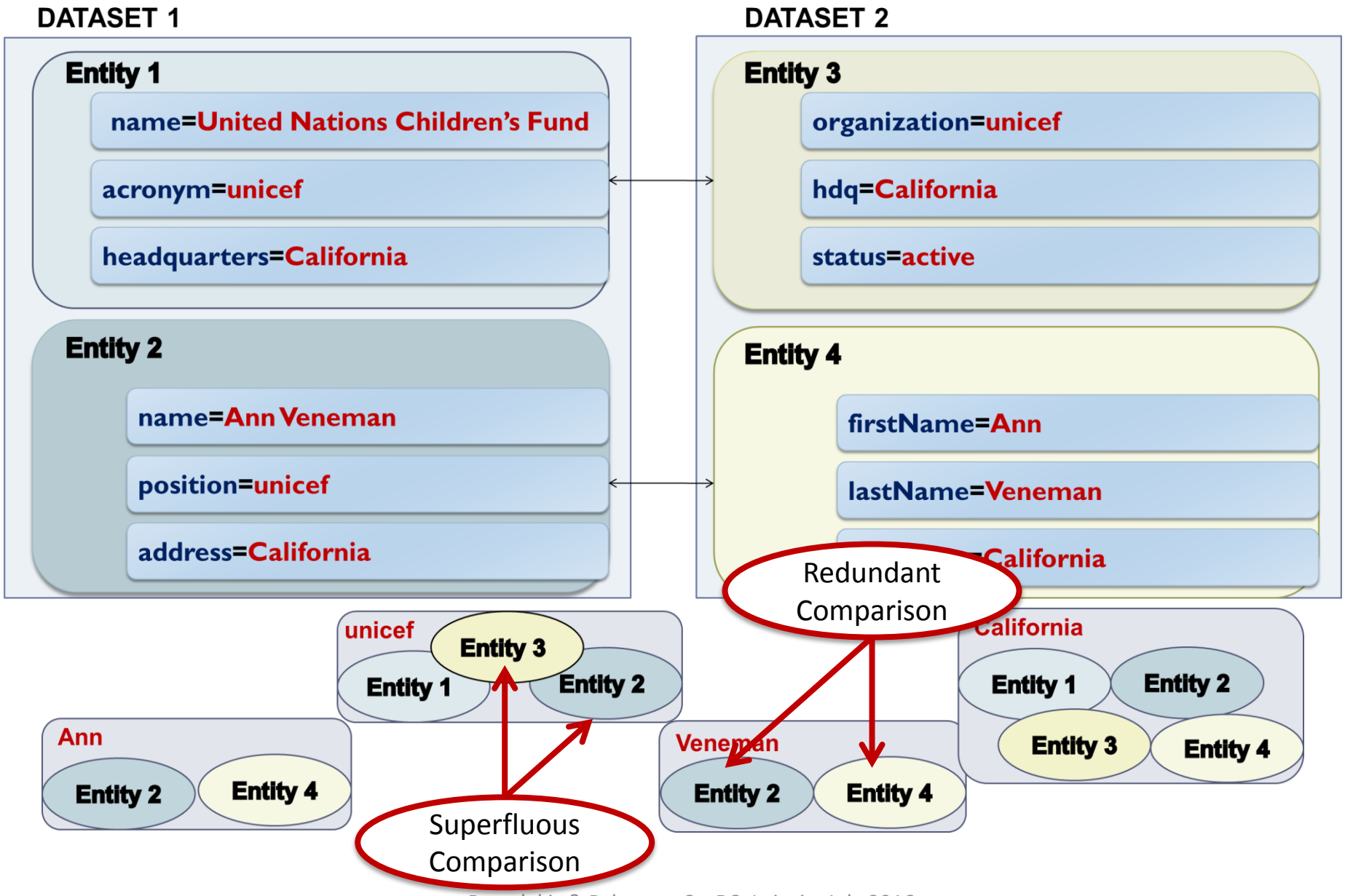
High Recall in the context of noisy entity profiles and extreme schema heterogeneity thanks to:

1. redundancy that reduces the likelihood of missed matches.
2. attribute-agnostic functionality that requires no schema semantics.

Low Precision because:

- the blocks are overlapping → redundant comparisons
- high number of comparisons between irrelevant entities → superfluous comparisons

Token Blocking Example



Part 4:

Block Processing Techniques

Outline

1. Introduction to Blocking
2. Blocking Methods for Relational Data
3. Blocking Methods for Web Data
4. Block Processing Techniques
   - Block Purging
   - Block Filtering
   - Block Clustering
   - Comparison Propagation
   - Iterative Blocking
5. Meta-blocking
6. Challenges
7. ER framework

General Principles

Goals:

1. eliminate *all* redundant comparisons
2. avoid *most* superfluous comparisons

without affecting matching comparisons (i.e., PC).

Depending on the granularity of their functionality, they are distinguished into:

1. Block-refinement
2. Comparison-refinement
   - Iterative Methods

Block Purging

Exploits power-law distribution of block sizes.

Targets **oversized blocks** (i.e., many comparisons, no duplicates)

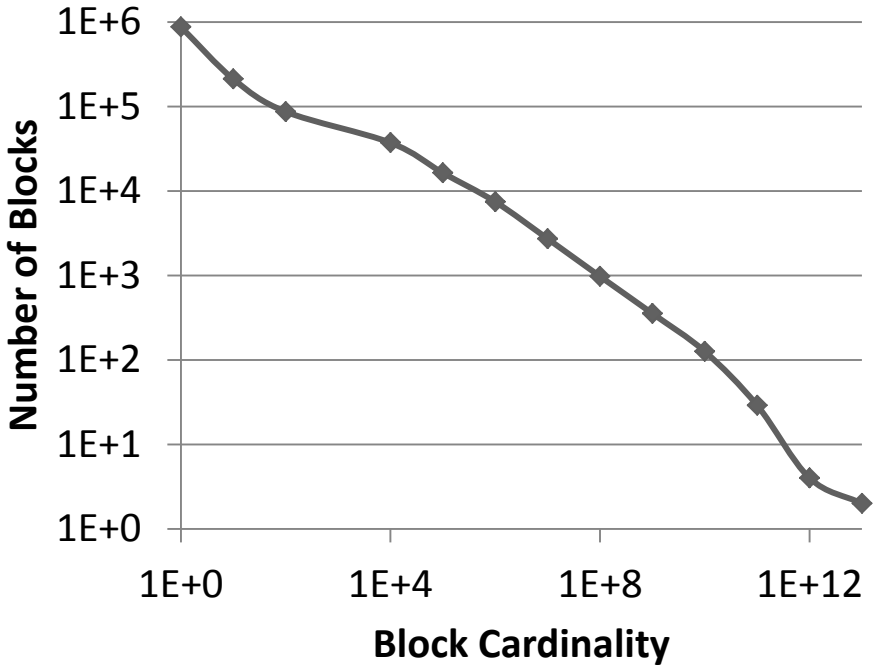
Discards them by setting an upper limit on:

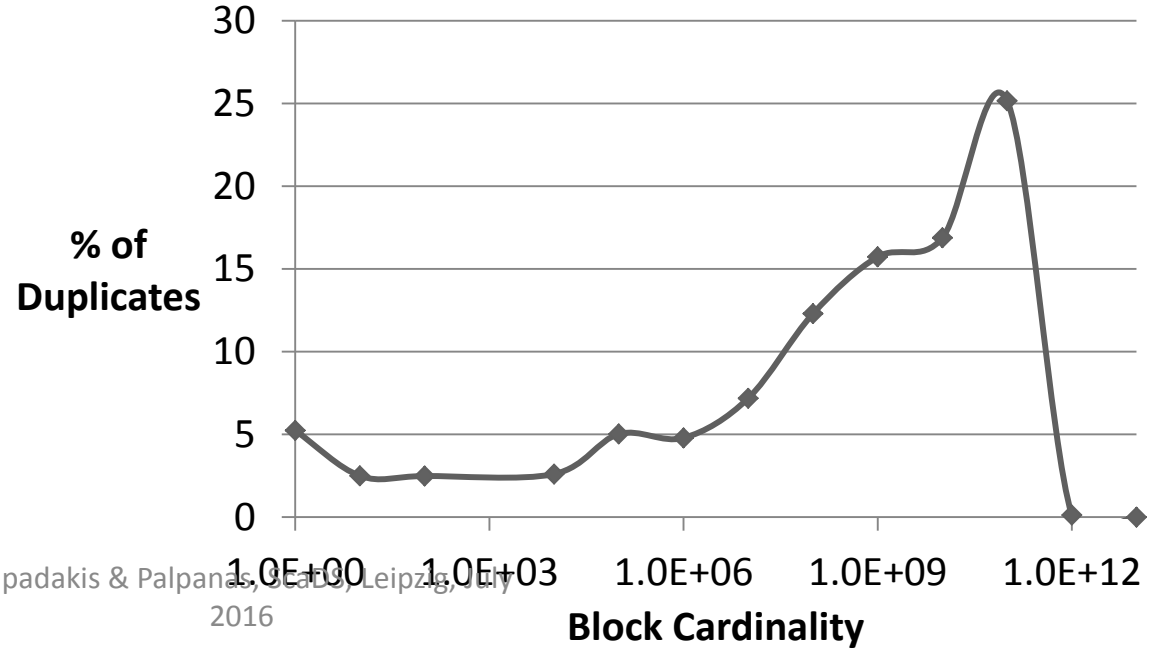
- the size of each block [Papadakis et al., WSDM 2011],
- the cardinality of each block [Papadakis et al., WSDM 2012]

Core method:

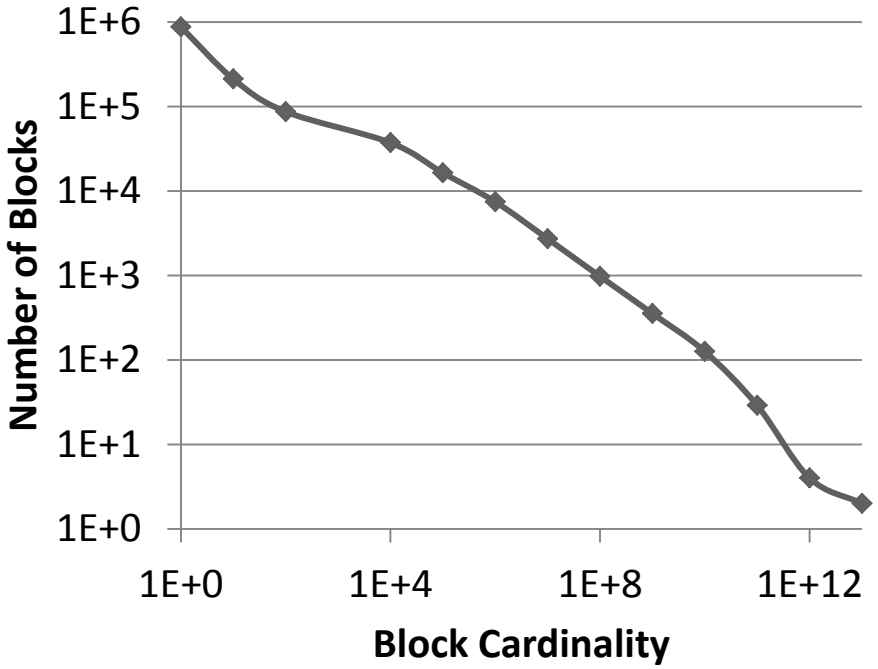
- Low computational cost.
- Low impact on effectiveness.
- Boosts efficiency to a large extent.

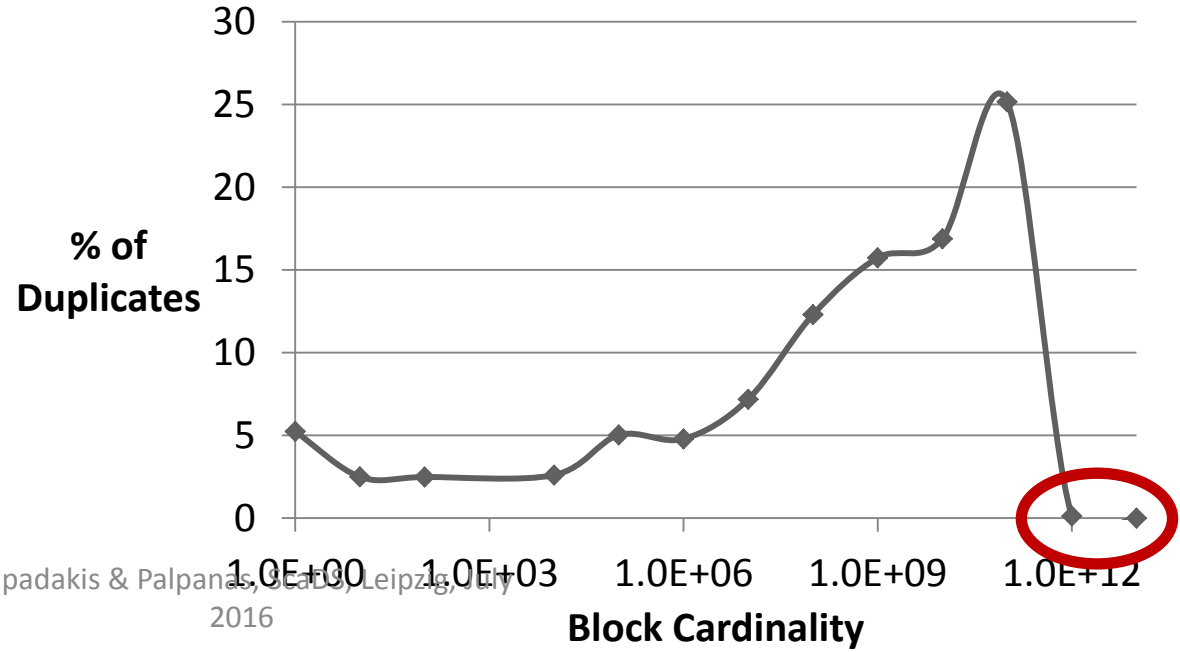
Distributions of Block Sizes and Duplicates



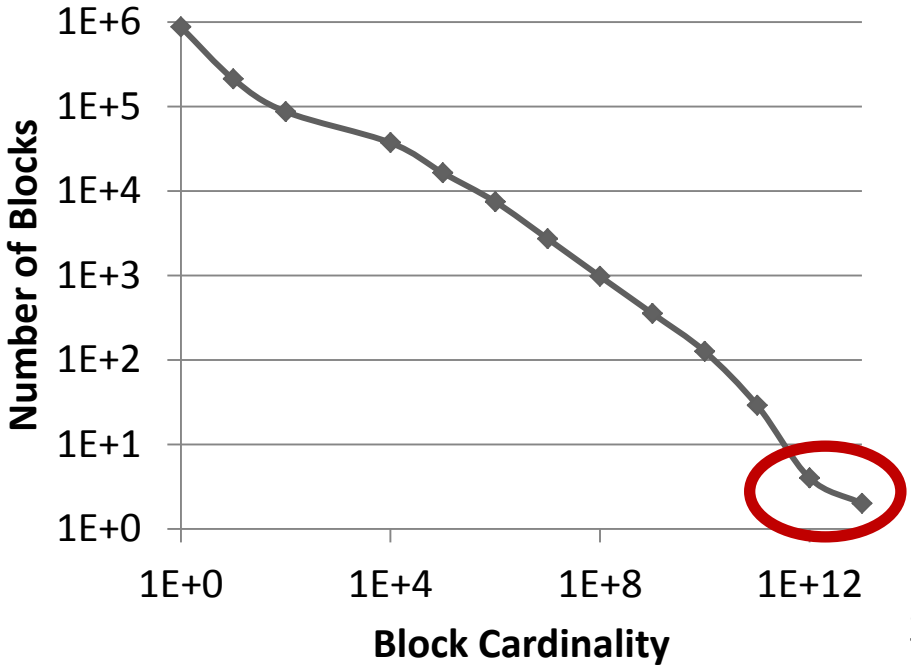


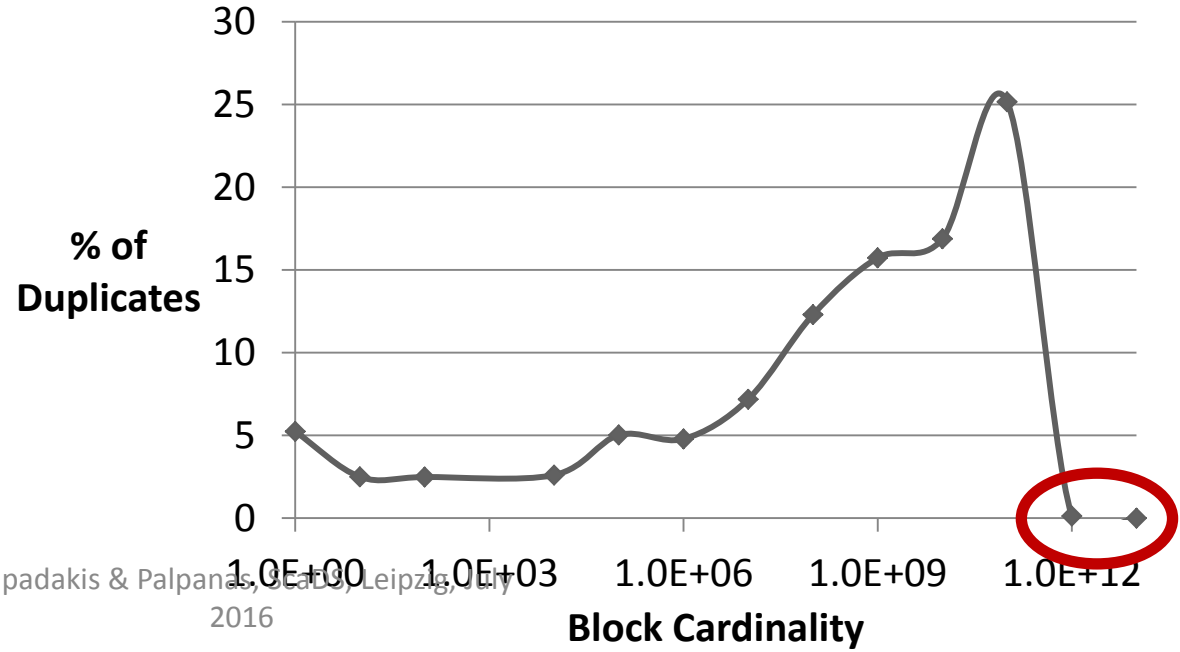
Distributions of Block Sizes and Duplicates





Distributions of Block Sizes and Duplicates





Block Filtering [Papadakis et. al, EDBT 2016]

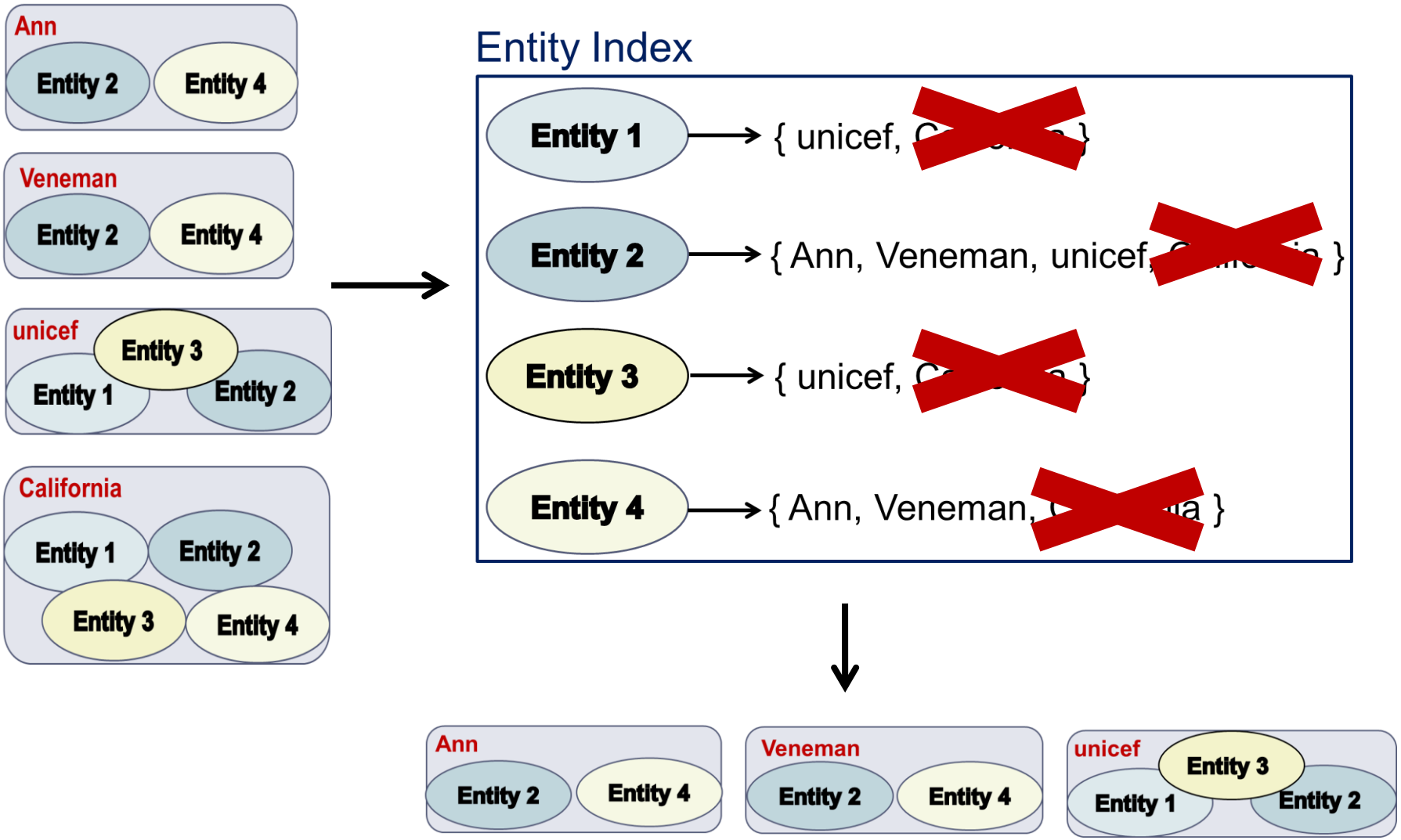
Main ideas:

- each block has a different importance for every entity it contains.
- Larger blocks are less likely to contain unique duplicates and, thus, are less important.

Algorithm

- sort blocks in ascending cardinality
- build Entity Index
- retain every entity in **r%** of its smallest blocks
- reconstruct blocks

Block Filtering Example



Block Clustering [Fisher et. al., KDD 2015]

Main idea:

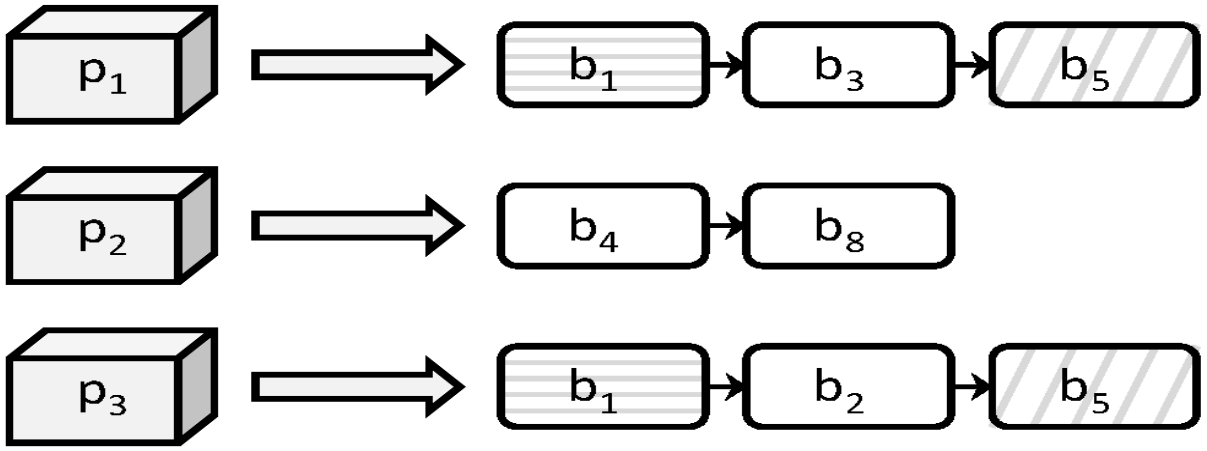
- restrict the size of every block into **[b_{min}, b_{max}]**
  - necessary in applications like privacy-preserving ER
  - operates so that ||B|| increases linearly with |E|

Algorithm

- recursive agglomerative clustering
  - merge similar blocks with size lower than b_{min}
  - split blocks with size larger than b_{max}
- until all blocks have the desired size

Comparison Propagation [Papadakis et al., JCDL 2011]

- Eliminate all redundant comparisons at no cost in recall.
- Naïve approach does not scale.
- Functionality:
  1. Build Entity Index
  2. Least Common Block Index condition.



Iterative Blocking [Whang et. Al, SIGMOD 2009]

Main idea:

integrate block processing with entity matching and reflect outcomes to subsequently processed blocks, until no new matches are detected.

Algorithm

- Put all blocks in a queue Q
- While Q is not empty
  - Get first block
  - Get matches with an ER algorithm (e.g., R-Swoosh)
    - For each new pair of duplicates $p_i \equiv p_j$
      - Merge their profiles $p'_i = p'_j =< p_i, p_j >$ and update them in all associated blocks
      - Place in Q all associated blocks that are not already in it

Part 5:

Meta-blocking

Motivation

DBPedia 3.0rc ↔ DBPedia 3.4

1.2 million entities ↔ 2.2 million entities

Motivation

DBPedia 3.0rc ↔ DBPedia 3.4

1.2 million entities ↔ 2.2 million entities

Brute-force approach

Comparisons: $2.58 \cdot 10^{12}$

Recall: 100%

Running time: 1,344 days → **3.7 years**

Motivation

DBPedia 3.0rc ↔ DBPedia 3.4

1.2 million entities ↔ 2.2 million entities

Brute-force approach

Comparisons: $2.58 \cdot 10^{12}$

Recall: 100%

Running time: 1,344 days → **3.7 years**

Token Blocking + Block Filtering + Comparison Propagation

Overhead time: <30 mins

Comparisons: $3.5 \cdot 10^{10}$

Recall: 99%

Total Running time: **19 days**

Motivation

DBPedia 3.0rc ↔ DBPedia 3.4

1.2 million entities ↔ 2.2 million entities

Brute-force approach

Comparisons: $2.58 \cdot 10^{12}$

Recall: 100%

Running time: 1,344 days → **3.7 years**

Token Blocking + Block Filtering + Comparison Propagation

Overhead time: <30 mins

Comparisons: $3.5 \cdot 10^{10}$

Recall: 99%

Total Running time: **19 days**

Motivation

DBPedia 3.0rc ↔ DBPedia 3.4

1.2 million entities ↔ 2.2 million entities

Brute-force approach

Comparisons: $2.58 \cdot 10^{12}$

Recall: 100%

Running time: 1,344 days → **3.7 years**

Token Blocking + Block Filtering + Comparison Propagation

Overhead time: <30 mins

Comparisons: $3.5 \cdot 10^{10}$

Recall: 99%

Total Running time: **19 days**

Token Blocking + Block Filtering + **??**

Meta-blocking [Papadakis et. al., TKDE 2014]

Goal:

restructure a **redundancy-positive** block collection into a new one that contains substantially lower number of redundant and superfluous comparisons, while maintaining the original number of matching ones ($\Delta PC \approx 0$, $\Delta PQ >> 1$) →

Meta-blocking [Papadakis et. al., TKDE 2014]

Goal:

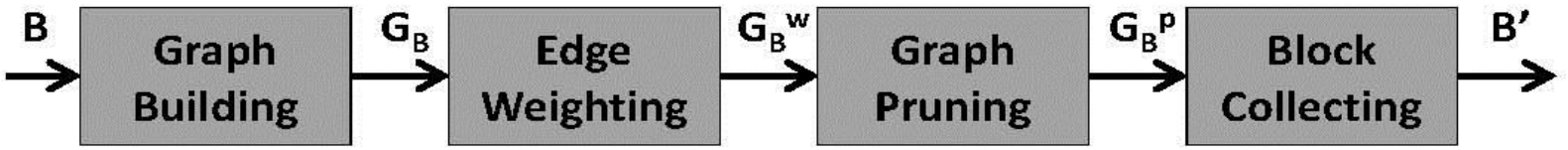
restructure a **redundancy-positive** block collection into a new one that contains substantially lower number of redundant and superfluous comparisons, while maintaining the original number of matching ones ($\Delta PC \approx 0$, $\Delta PQ >> 1$) →

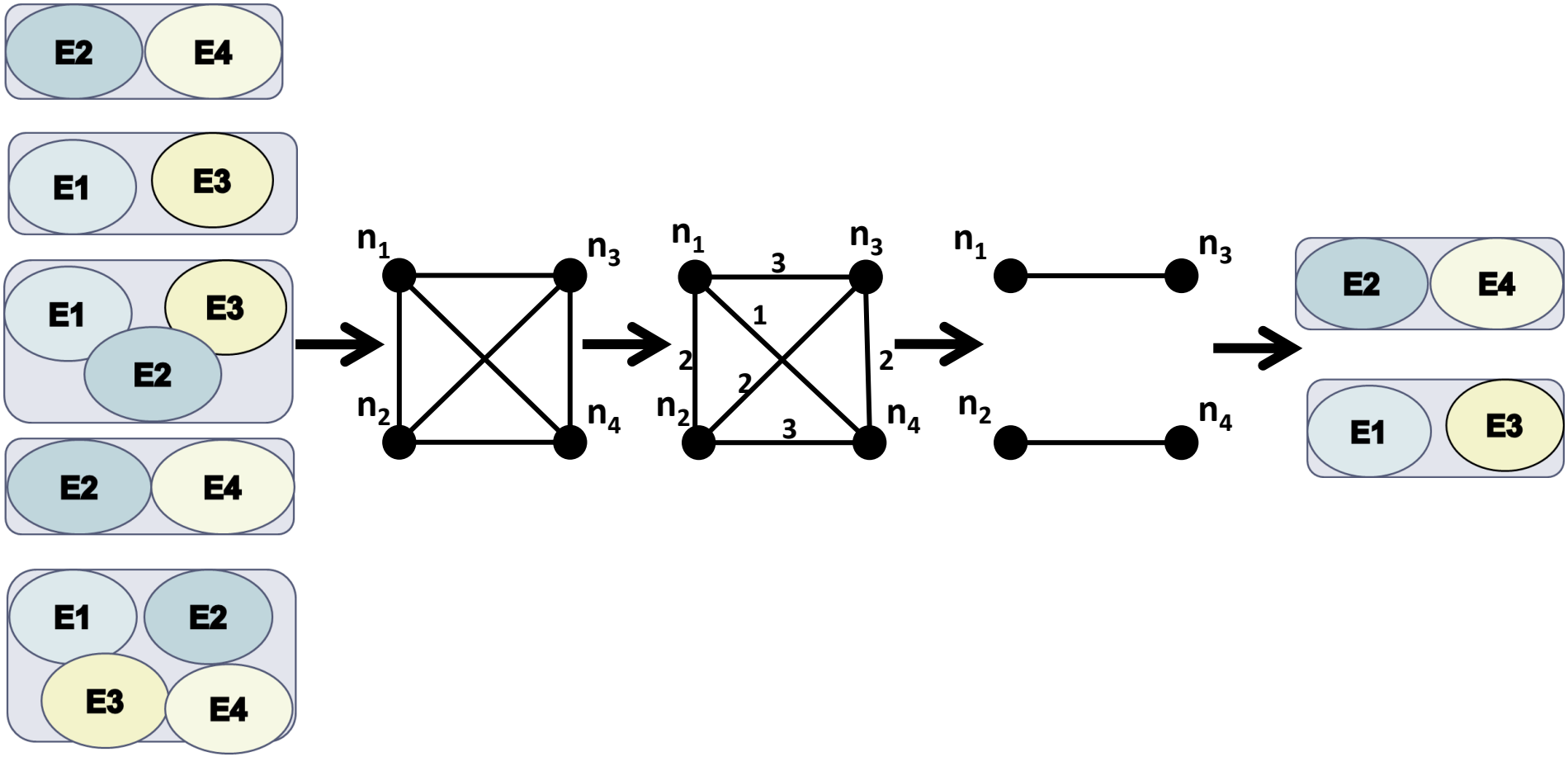
Main idea:

common blocks provide valuable evidence for the similarity of entities

→ the more blocks two entities share, the more similar and the more likely they are to be matching

Outline of Meta-blocking





Graph Building

For every block:

- for every entity → add a node
- for every pair of co-occurring entities → add an undirected edge

Blocking graph:

- It eliminates all redundant comparisons →
  no parallel edges.
- Low materialization cost →
  implicit materialization through inverted indices
- Different from similarity graph!

Edge Weighting

Five generic, attribute-agnostic weighting schemes that rely on the following evidence:

- the number of blocks shared by two entities
- the size of the common blocks
- the number of blocks or comparisons involving each entity.

Computational Cost:

- In theory, equal to executing all pair-wise comparisons in the given block collection.
- In practice, significantly lower because it does not employ string similarity metrics.

Weighting Schemes

1. Aggregate Reciprocal Comparisons Scheme (ARCS)

$$w_{ij} = \sum_{b_k \in B_{ij}} \frac{1}{||b_k||}$$

2. Common Blocks Scheme (CBS)

$$w_{ij} = |B_{ij}|$$

3. Enhanced Common Blocks Scheme (ECBS)

$$w_{ij} = |B_{ij}| \cdot \log \frac{|B|}{|B_i|} \cdot \log \frac{|B|}{|B_j|}$$

4. Jaccard Scheme (JS)

$$w_{ij} = \frac{|B_{ij}|}{|B_i| + |B_j| - |B_{ij}|}$$

5. Enhanced Jaccard Scheme (EJS)

$$w_{ij} = \frac{|B_{ij}|}{|B_i| + |B_j| - |B_{ij}|} \cdot \log \frac{|V_G|}{|v_i|} \cdot \log \frac{|V_G|}{|v_j|}$$

Graph Pruning

Pruning algorithms

1. Edge-centric
2. Node-centric

they produce directed blocking graphs

Pruning criteria

Scope:

1. Global
2. Local

Functionality:

1. Weight thresholds
2. Cardinality thresholds

Edge-centric

scope	functionality: weight	functionality: cardinality
global	WEP	CEP
local	x	x

(a)

Node-centric

scope	functionality: weight	functionality: cardinality
global	x	CNP
local	WNP	CNP

(b)

Thresholds for Graph Pruning

Experiments show robust behavior of the following configurations:

1. **Weighted Edge Pruning** (WEP)
   threshold: average weight across all edges
2. **Cardinality Edge Pruning** (CEP)
   threshold: K = BPE·|E|/2
3. **Weighted Node Pruning** (WNP)
   threshold: for each node, the average weight of the adjacent edges
4. **Cardinality Node Pruning** (CNP)
   threshold: for each node, k=BPE-1

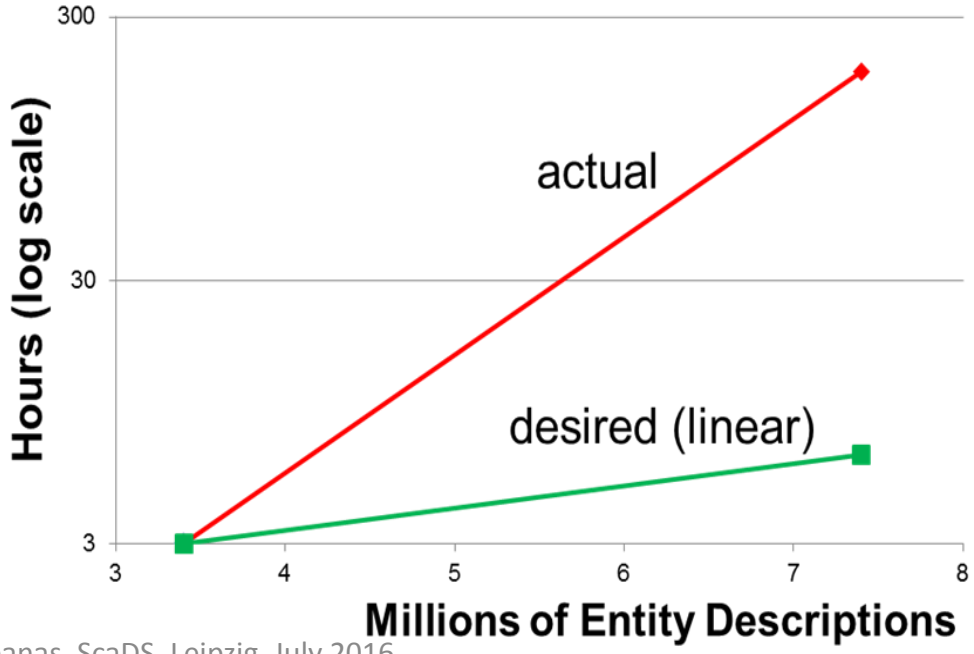
Meta-blocking Challenges

1. Time Efficiency

- Bottleneck: edge weighting
- Depends on $||B||$, BPE
  - $|E| = 3.4 \times 106$, $||B|| = 4 \times 1010$, BPE=15 → 3 hours
  - $|E| = 7.4 \times 106$, $||B|| = 2 \times 1011$, BPE=40 → 186 hours

2. Effectiveness

Simple pruning rules



Enhancing Meta-blocking Efficiency

- Block Filtering
  - r = 0.8 → 4 times faster processing, on average
  - reduces both ||B|| and BPE

- Optimized Edge Weighting
  [Papadakis et. al., EDBT 2016]
  - Entity-based instead of Block-based implementation
  - An order of magnitude faster processing, in combination with Block Filtering

- Parallel Meta-blocking
  [Efthymiou et. al., BigData 2015]
  - Load-balanced, distributed approach based on MapReduce (Apache Hadoop)

Motivation

DBPedia 3.0rc ↔ DBPedia 3.4

Brute-force approach

Comparisons: $2.58 \cdot 10^{12}$

Recall: 100%

Running time: 1,344 days → **3.7 years**

Token Blocking + Block Filtering + Comparison Propagation

Overhead time: <30 mins

Comparisons: $3.5 \cdot 10^{10}$

Recall: 99%

Total Running time: **19 days**

Token Blocking + Block Filtering + **Meta-blocking**

Motivation

DBPedia 3.0rc ↔ DBPedia 3.4

Brute-force approach

Comparisons: $2.58 \cdot 10^{12}$

Recall: 100%

Running time: 1,344 days → **3.7 years**

Token Blocking + Block Filtering + Comparison Propagation

Overhead time: <30 mins

Comparisons: $3.5 \cdot 10^{10}$

Recall: 99%

Total Running time: **19 days**

Token Blocking + Block Filtering + **Meta-blocking**

Overhead time: 4 hours

Comparisons: $8.95 \cdot 10^{6}$

Recall: 99%

Total Running time: **10 hours**

Motivation

DBPedia 3.0rc ↔ DBPedia 3.4

Brute-force approach

Comparisons: $2.58 \cdot 10^{12}$

Recall: 100%

Running time: 1,344 days → **3.7 years**

Token Blocking + Block Filtering + Comparison Propagation

Overhead time: <30 mins

Comparisons: $3.5 \cdot 10^{10}$

Recall: 99%

Total Running time: **19 days**

Token Blocking + Block Filtering + **Meta-blocking**

Overhead time: 4 hours

Comparisons: $8.95 \cdot 10^{6}$

Recall: 99%

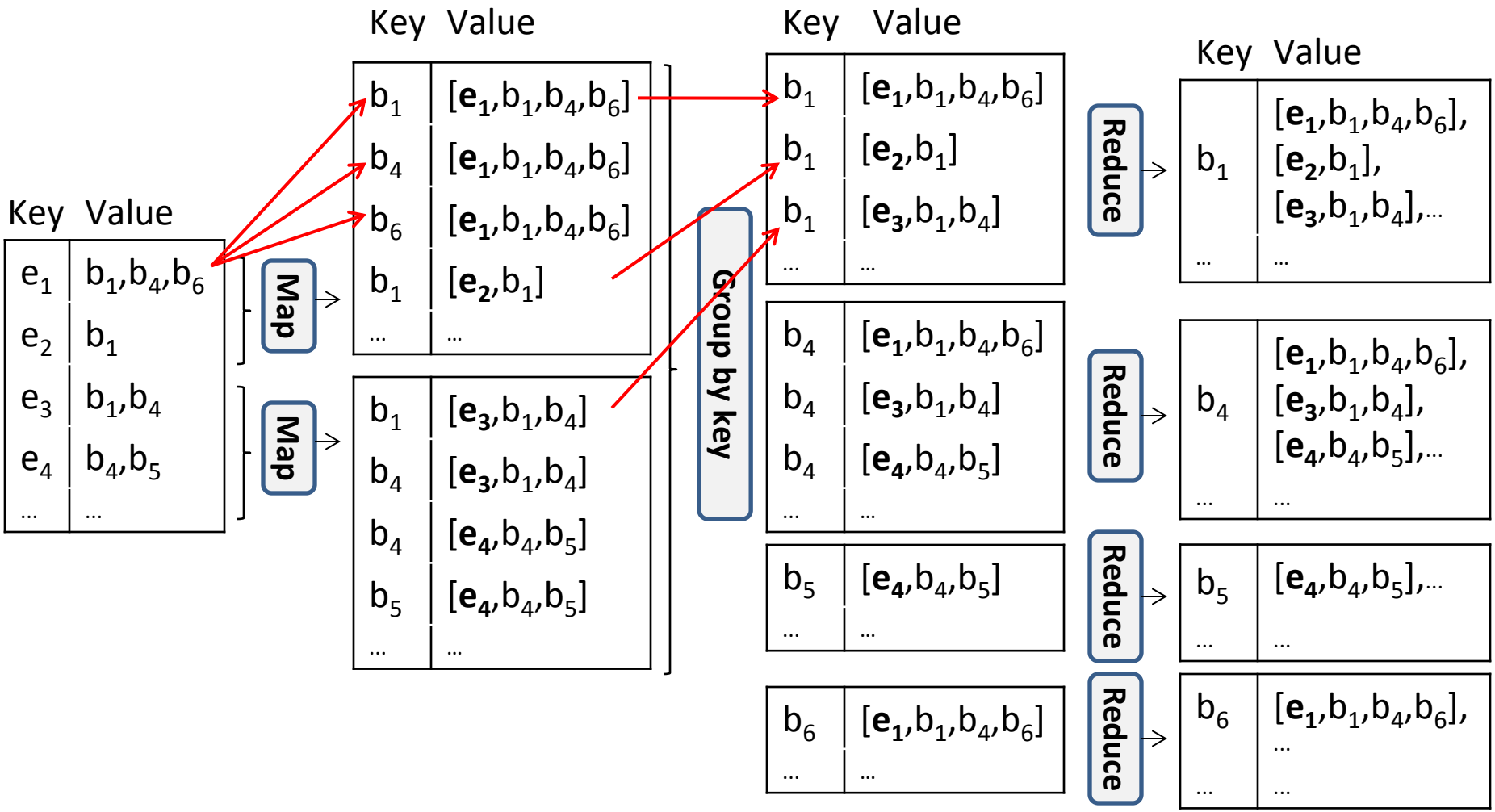
Total Running time: **10 hours**

Parallel Meta-blocking

- Two strategies:
  - **Basic:** explicitly creates the blocking graph
    - it performs all weight computations and stores all edges in disk
  - **Advanced:** uses the blocking graph as a conceptual model
    - enriches the input of the pruning algorithms with all the information necessary to compute the weights

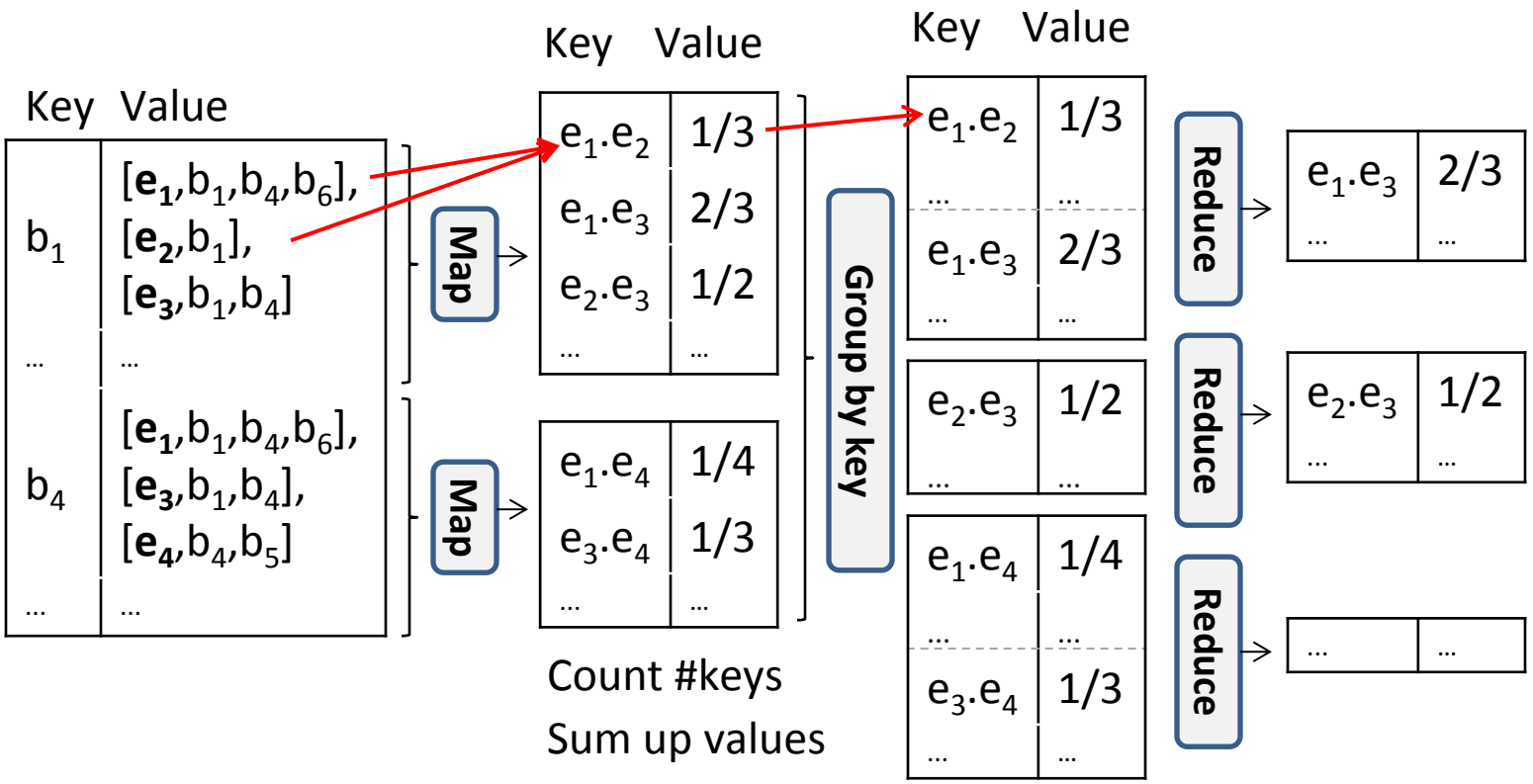
Meta-blocking (advanced)

Pre-processing



Meta-blocking (advanced)

Weighted Edge Pruning (WEP) & Jaccard Scheme (JS)



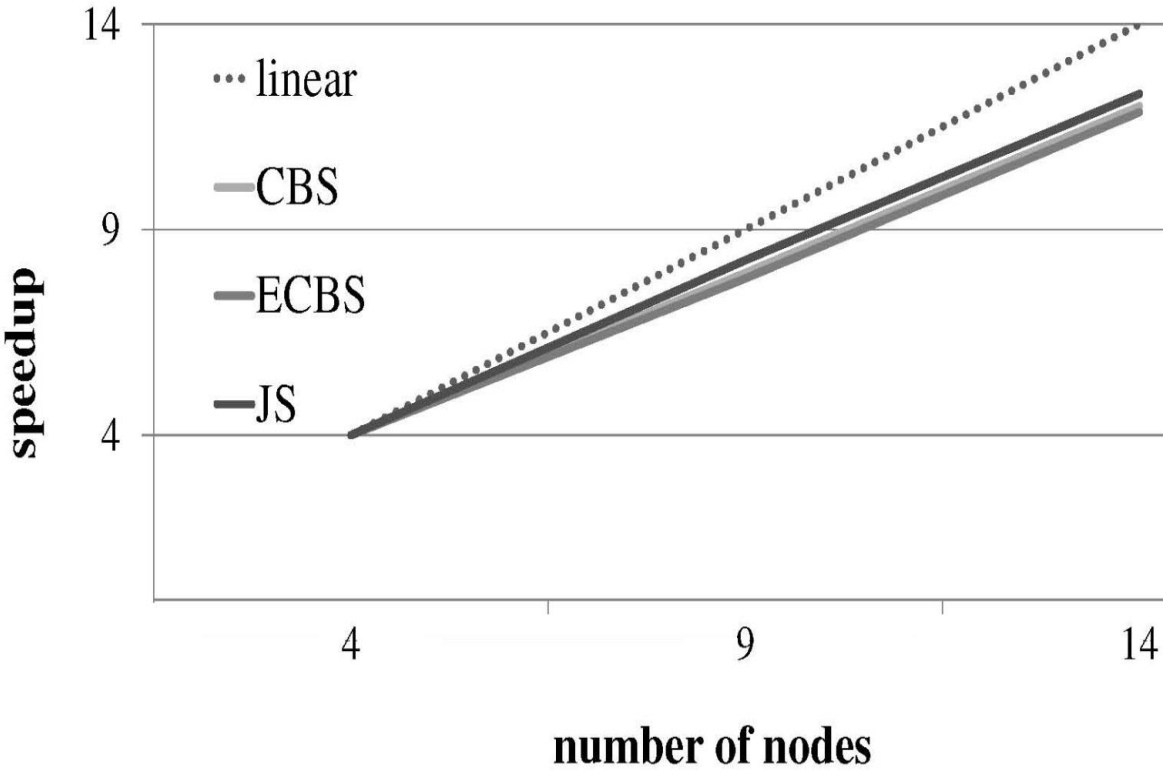
Meta-blocking (advanced)

Weighted Edge Pruning (WEP) & Jaccard Scheme (JS)

		DB_C	
		Basic	**Adv.**
Block Filtering		2	2
CEP	CBS	222	22
	ECBS	240	38
	JS	223	28
CNP	CBS	491	301
	ECBS	555	383
	JS	534	363
WEP	CBS	220	38
	ECBS	219	46
	JS	219	41
WNP	CBS	498	304
	ECBS	568	389
	JS	553	373

Meta-blocking (advanced)

Weighted Edge Pruning (WEP) & Jaccard Scheme (JS)



Parallel Meta-blocking achieves linear scale-up!

Enhancing Meta-blocking Effectiveness

Supervised Meta-blocking [Papadakis et. al., VLDB 2014]

Goal:

more accurate and comprehensive methodology for pruning the edges of the blocking graph.

Solution:

- model edge pruning as a classification task per edge
- two classes: “likely match”, “unlikely match”
- associate each edge with a set of features that are:
  - generic
  - effective
  - efficient
  - minimal

Enhancing Meta-blocking Effectiveness Feature Engineering

	source of evidence			target			complexity		scope		
	block-based	graph-based	iterative	edge-specific	node-specific	hybrid	raw	derived	local	global	hybrid
CF_IBF	✓					✓		✓			✓
Jaccard_Sim	✓			✓				✓	✓		
RACCB	✓			✓				✓	✓		
Node_Degree		✓			✓		✓		✓		
Iterative_Degree			✓		✓			✓		✓	
Transitive_Degree		✓			✓			✓		✓	

Examined all 63 possible combinations to find the minimal set of features, which comprises the first four features.

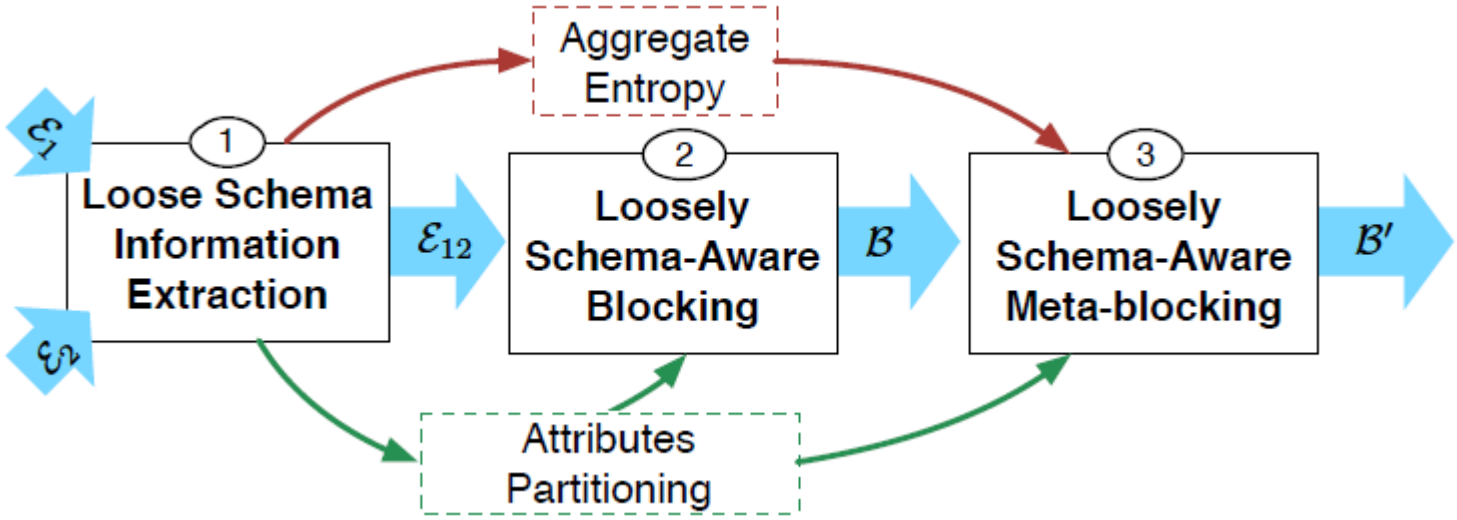
We combined them with state-of-the-art classification algorithms:

C4.5, SVM, Naïve Bayes, Bayesian Networks.

Robust performance w.r.t. algorithm parameters.

BLAST: Loosely Schema-aware Meta-blocking
[Simonini et. al., VLDB 2017]

- Goal:
  improve the edge weighting and pruning in unsupervised WNP with loose schema information
- Solution:



It works for Dirty ER, as well.

BLAST Algorithm

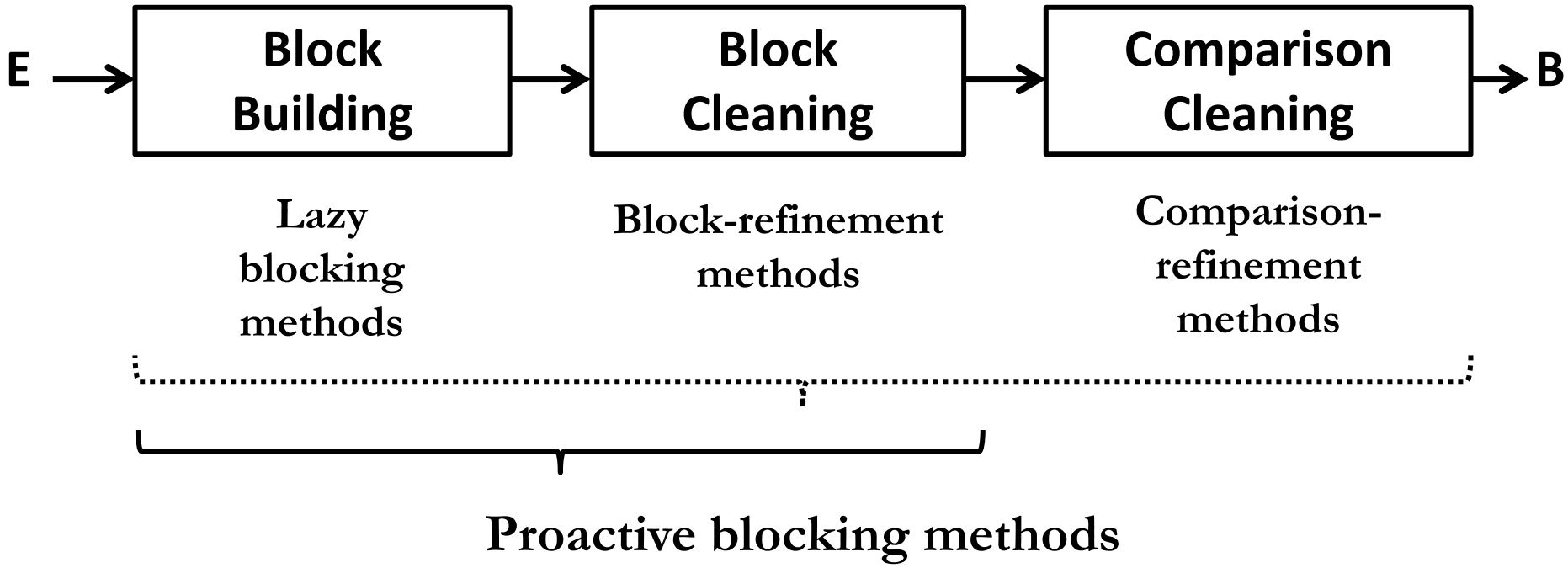
1. Attributes Partitioning accelerates **Attribute Clustering** by using LSH for token-based Jaccard similarity between attribute names
2. BLAST improves **edge weighting** based on the following relationships:
   every edge → several blocking keys (tokens) → multiple attribute names → aggregate entropy · Pearson's χ^2
3. BLAST improves **edge pruning** in two ways:
   1. Local weight threshold independent of the size of each node neighborhood (i.e., number of edges):

$$\theta_i = \frac{argmax_i\, w(e_{ij})}{2}$$

   2. An edge e_{ij} is retained if $w(e_{ij}) \geq \frac{\theta_i + \theta_j}{2}$.

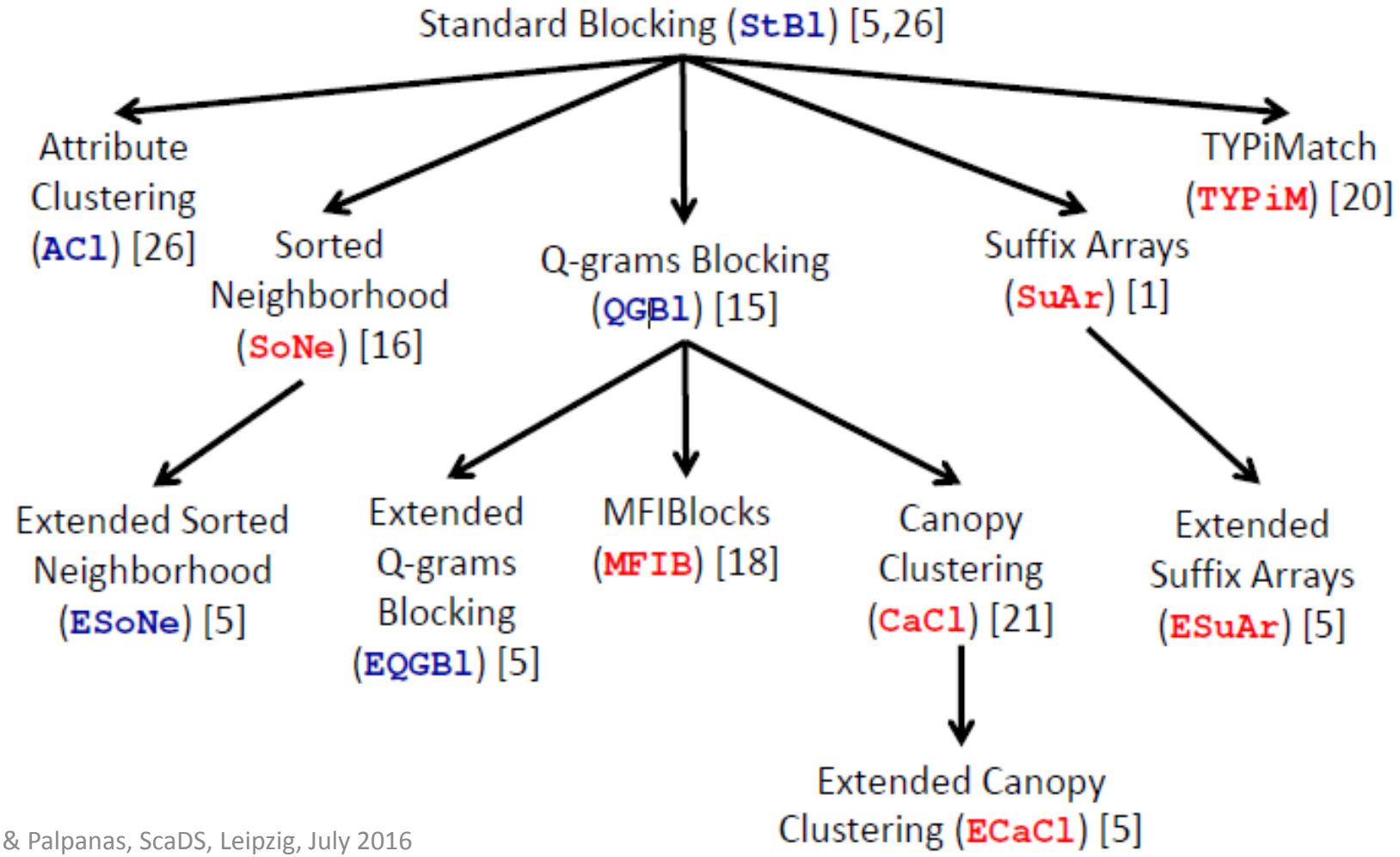
Comparative Analysis of Approximate Blocking Techniques [Papadakis et. al., VLDB 2016]

- employed 3 sub-tasks of blocking



Comparative Analysis of Approximate Blocking Techniques [Papadakis et. al., VLDB 2016]

- considered 5 lazy and 7 proactive blocking methods



Experimental Analysis Setup

- Block Cleaning methods:
  1. Block Purging
  2. Block Filtering

- Comparison Cleaning methods:
  1. Comparison Propagation
  2. Iterative Blocking
  3. Meta-blocking

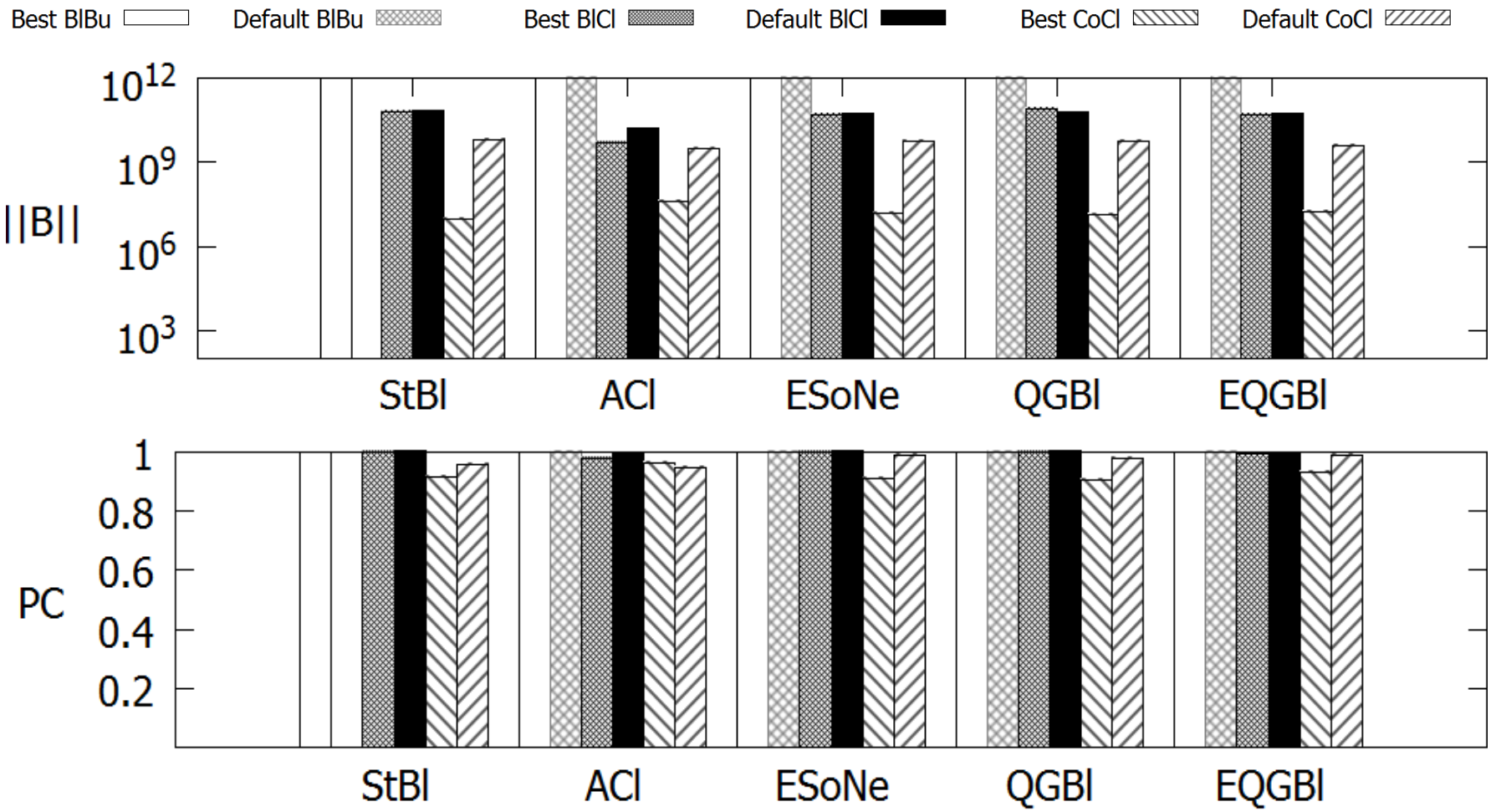
Experimental Analysis Setup

- Exhaustive parameter tuning to identify two configurations for each method:
  1. Best configuration per dataset → maximizes
  $$\boldsymbol{a}(\boldsymbol{B},\boldsymbol{E}) = \boldsymbol{RR}(\boldsymbol{B},\boldsymbol{E}) \cdot \boldsymbol{PC}(\boldsymbol{B},\boldsymbol{E})$$
  2. Default configuration → highest average $\boldsymbol{a}$ across all datasets

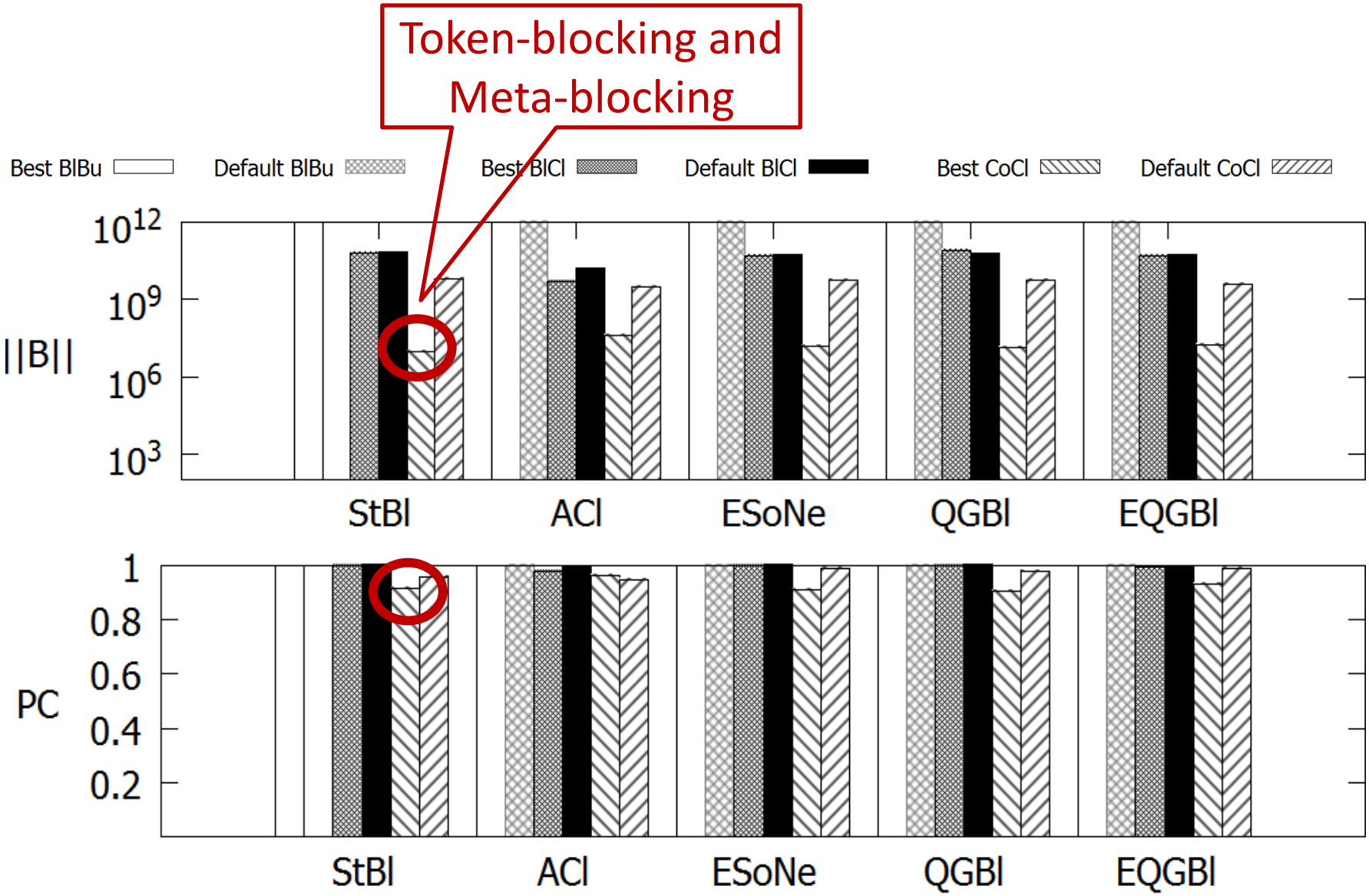
- Extensive experiments measuring effectiveness and time efficiency over **5 real** datasets (up to 3.3M entities).

- Scalability analysis over **7 synthetic** datasets (up to 2M entities).

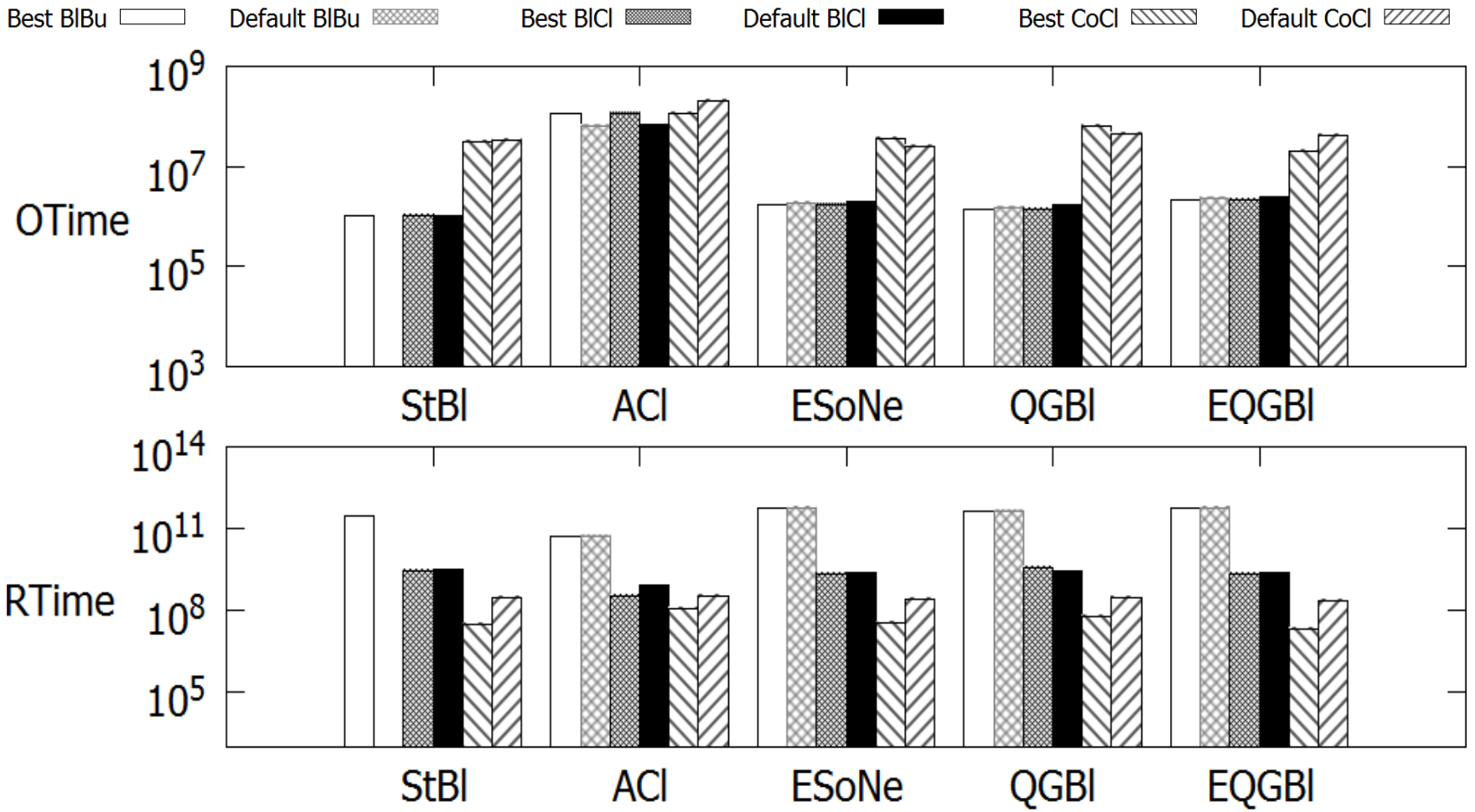
Effectiveness of Lazy Methods on DBPedia



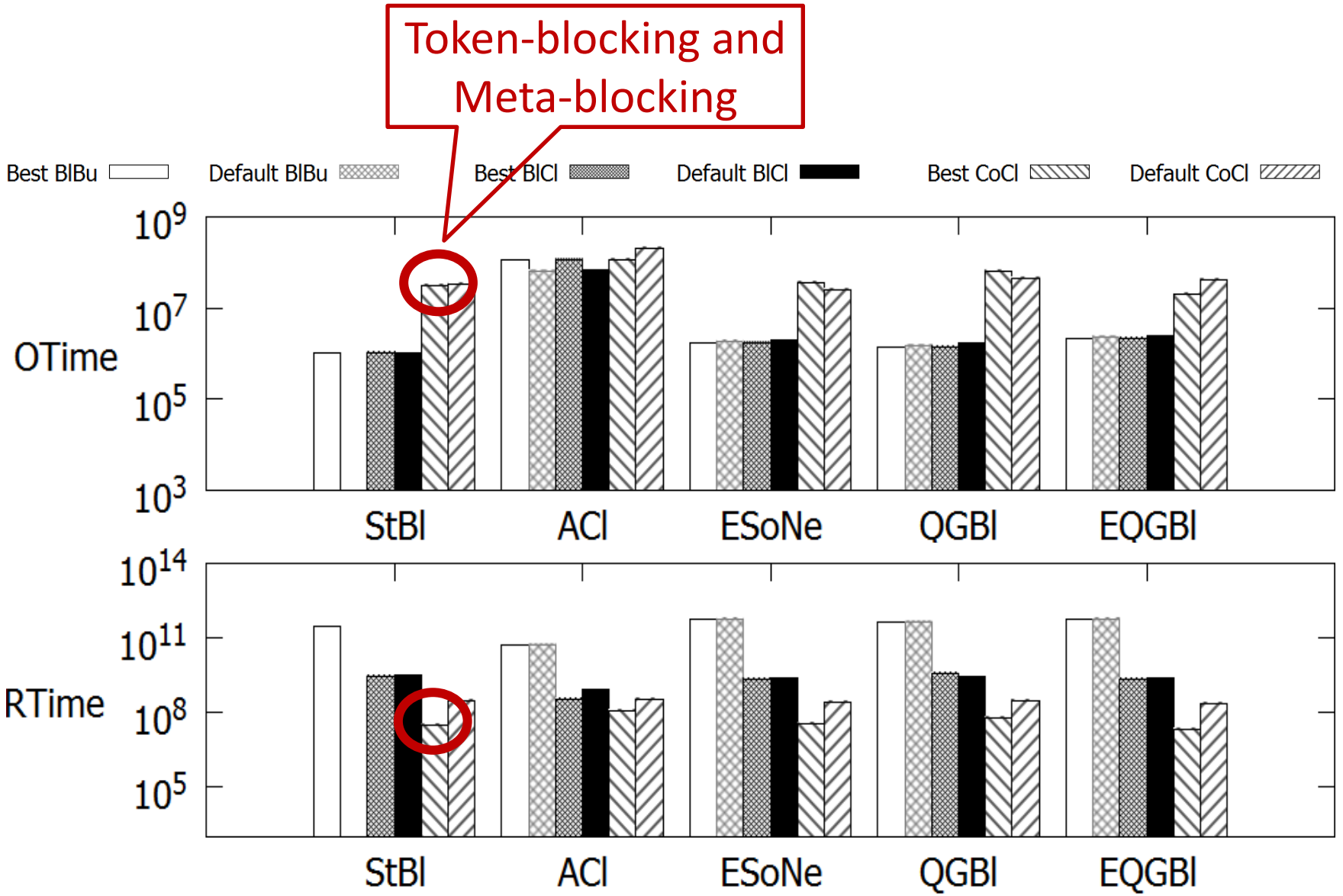
Effectiveness of Lazy Methods on DBPedia



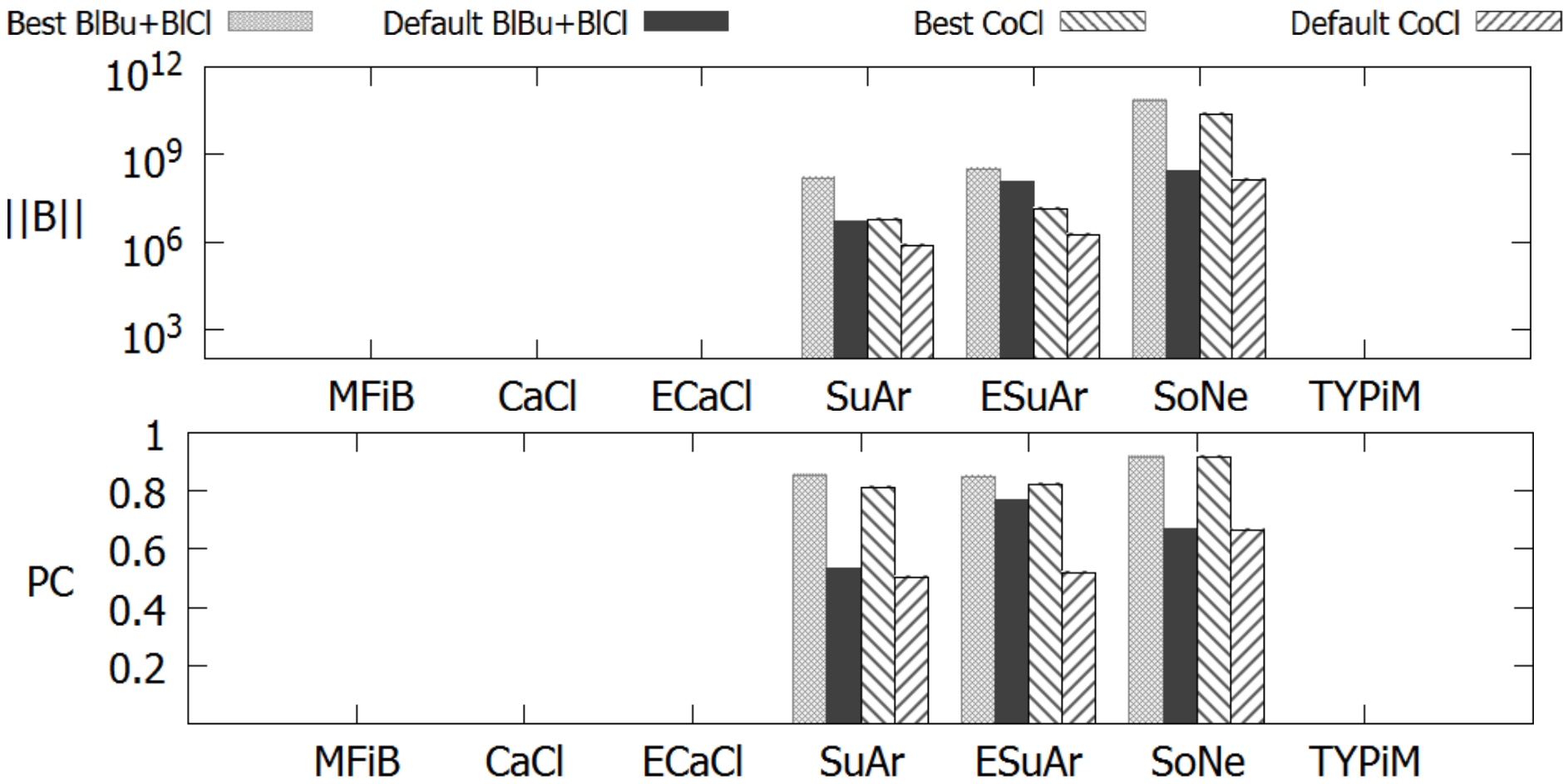
Time Efficiency of Lazy Methods on DBPedia



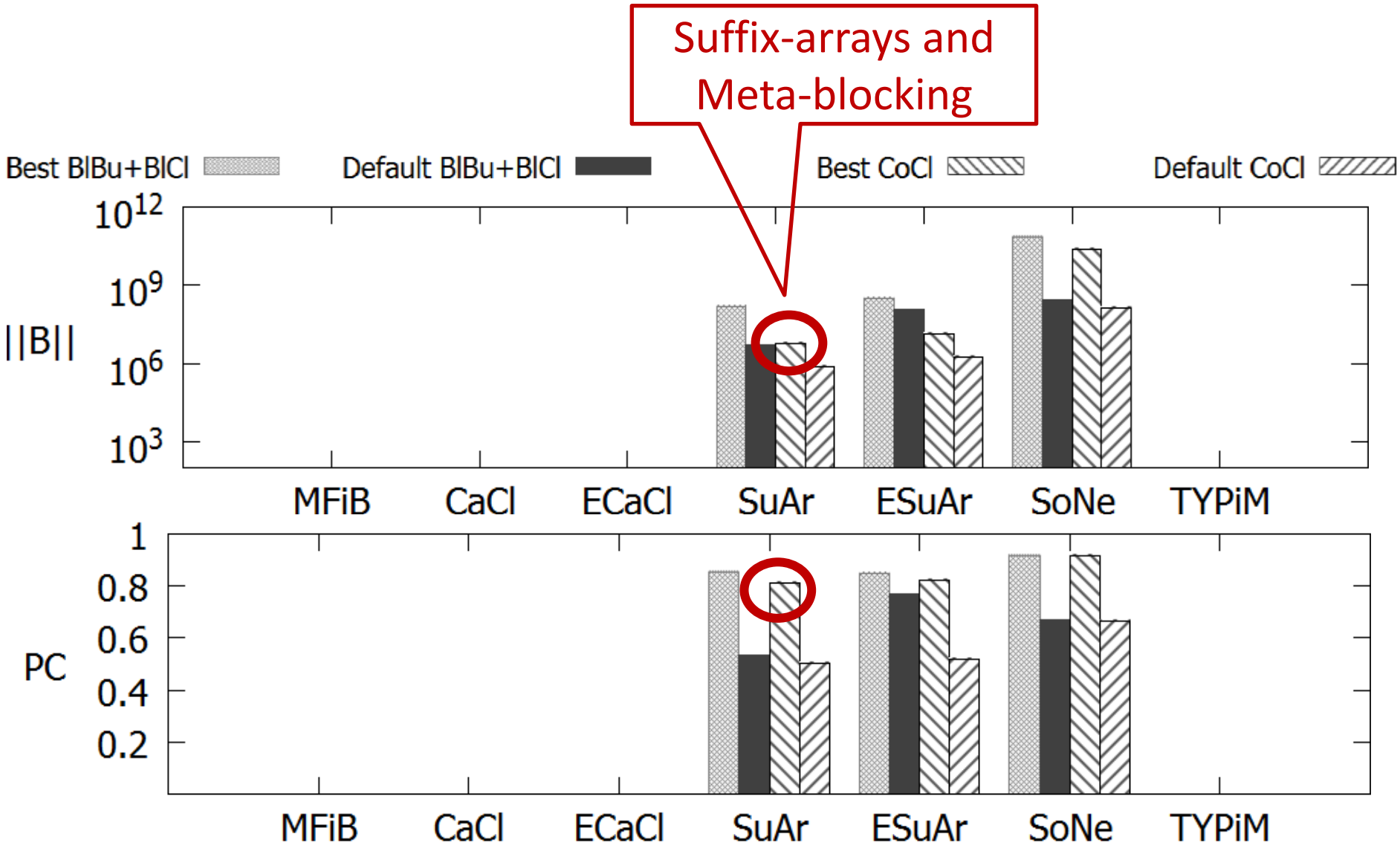
Time Efficiency of Lazy Methods on DBPedia



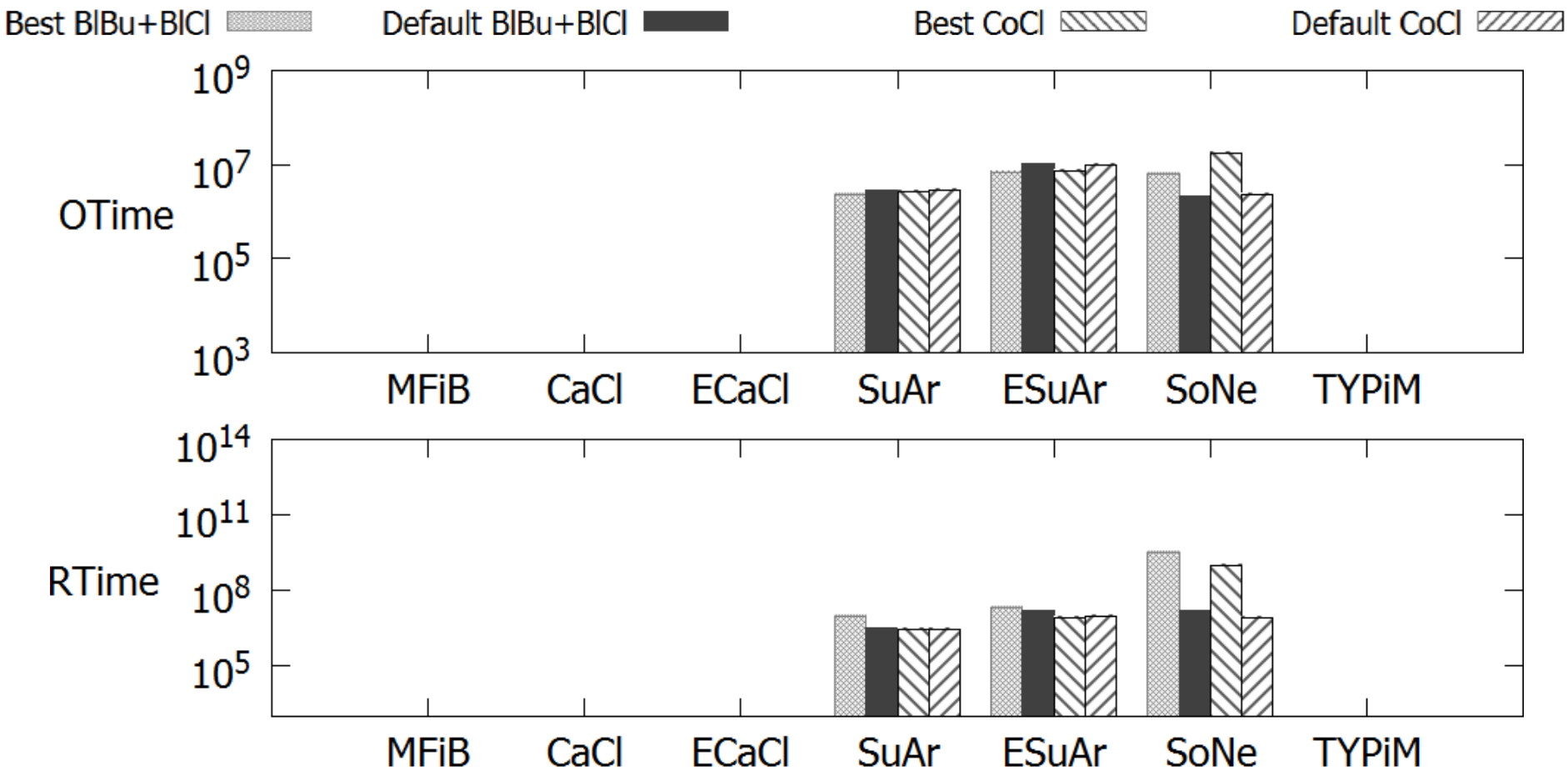
Effectiveness of Proactive methods on DBPedia



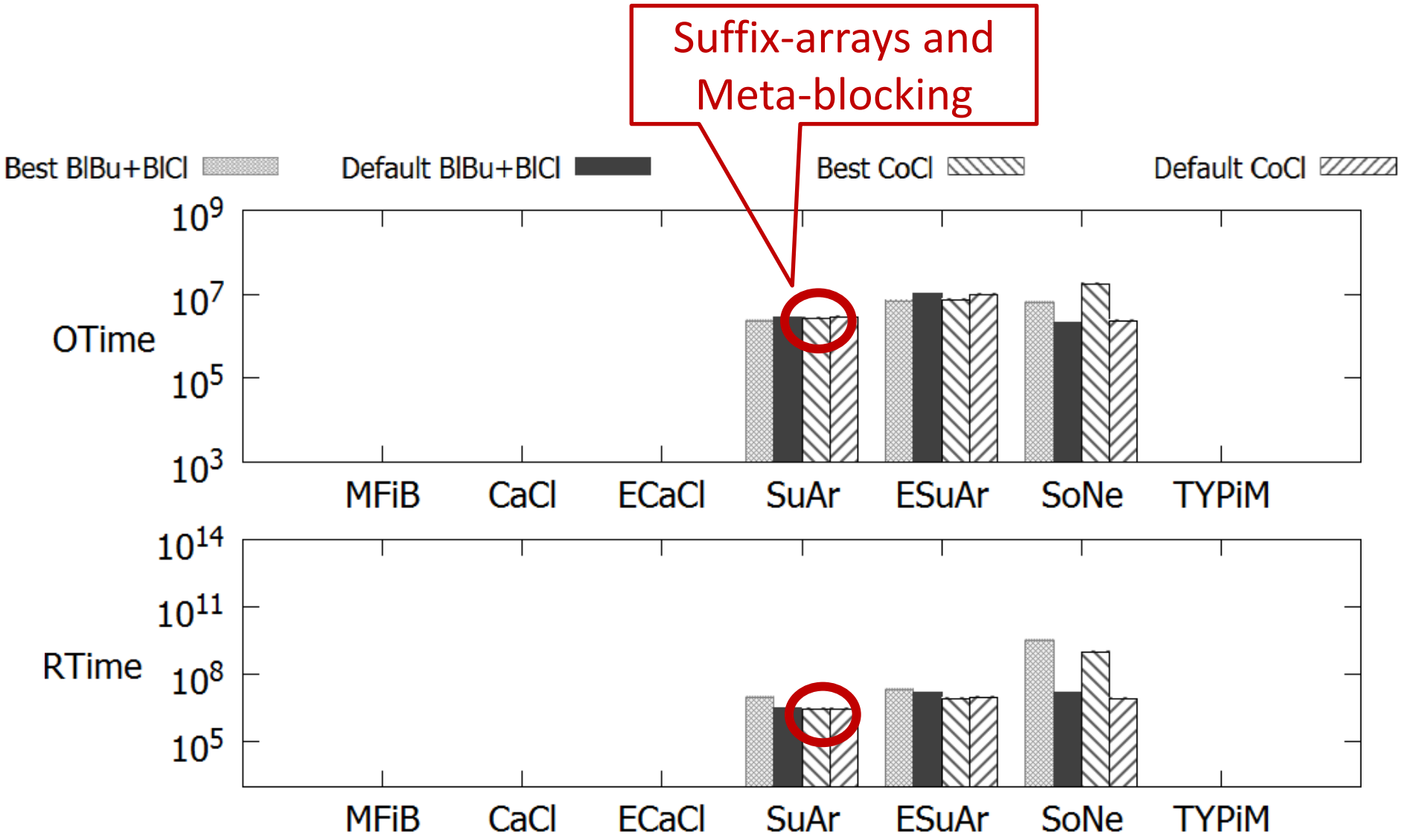
Effectiveness of Proactive methods on DBPedia



Time Efficiency of Proactive Methods on DBPedia



Time Efficiency of Proactive Methods on DBPedia



Part 6:

Challenges

Automatic Configuration

Facts:

- Several parameters in every blocking workflow
  - Both for lazy and proactive methods
- Blocking performance sensitive to internal configuration
  - Experimentally verified in [Papadakis et. al., VLDB 2016]
- Manual fine-tuning required

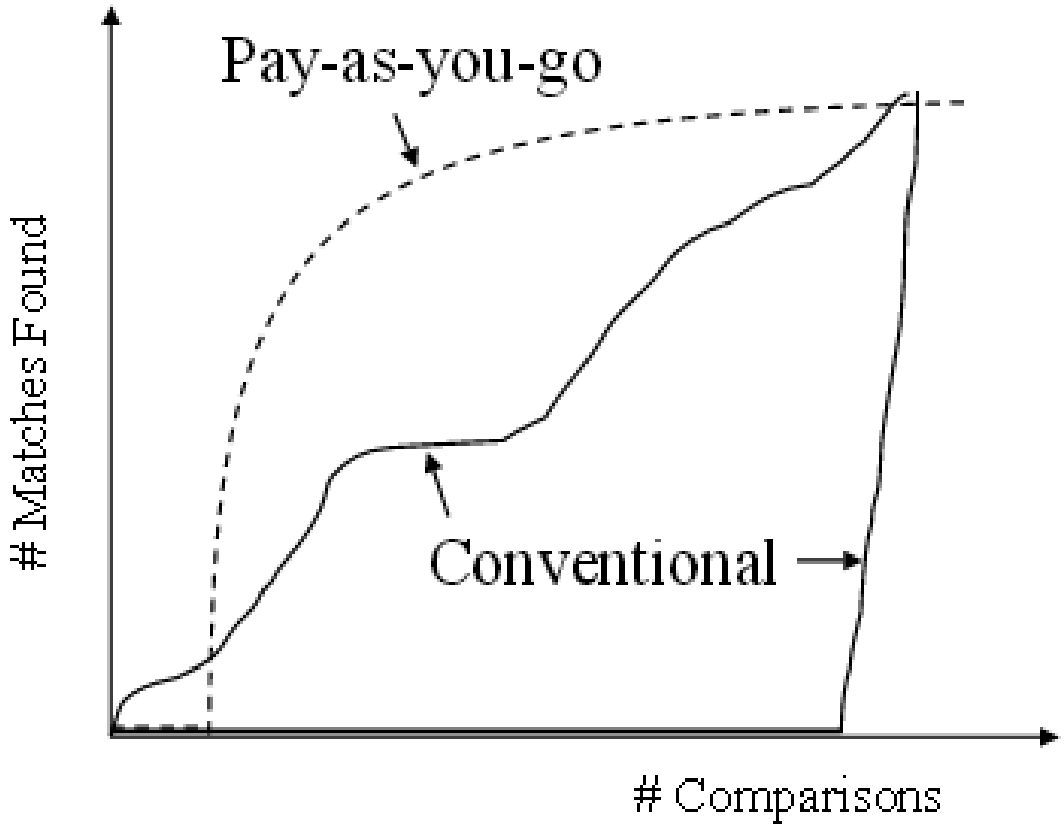
Open Research Directions:

- Plug-and-play blocking
- Data-driven configuration

Progressive Blocking

Facts:

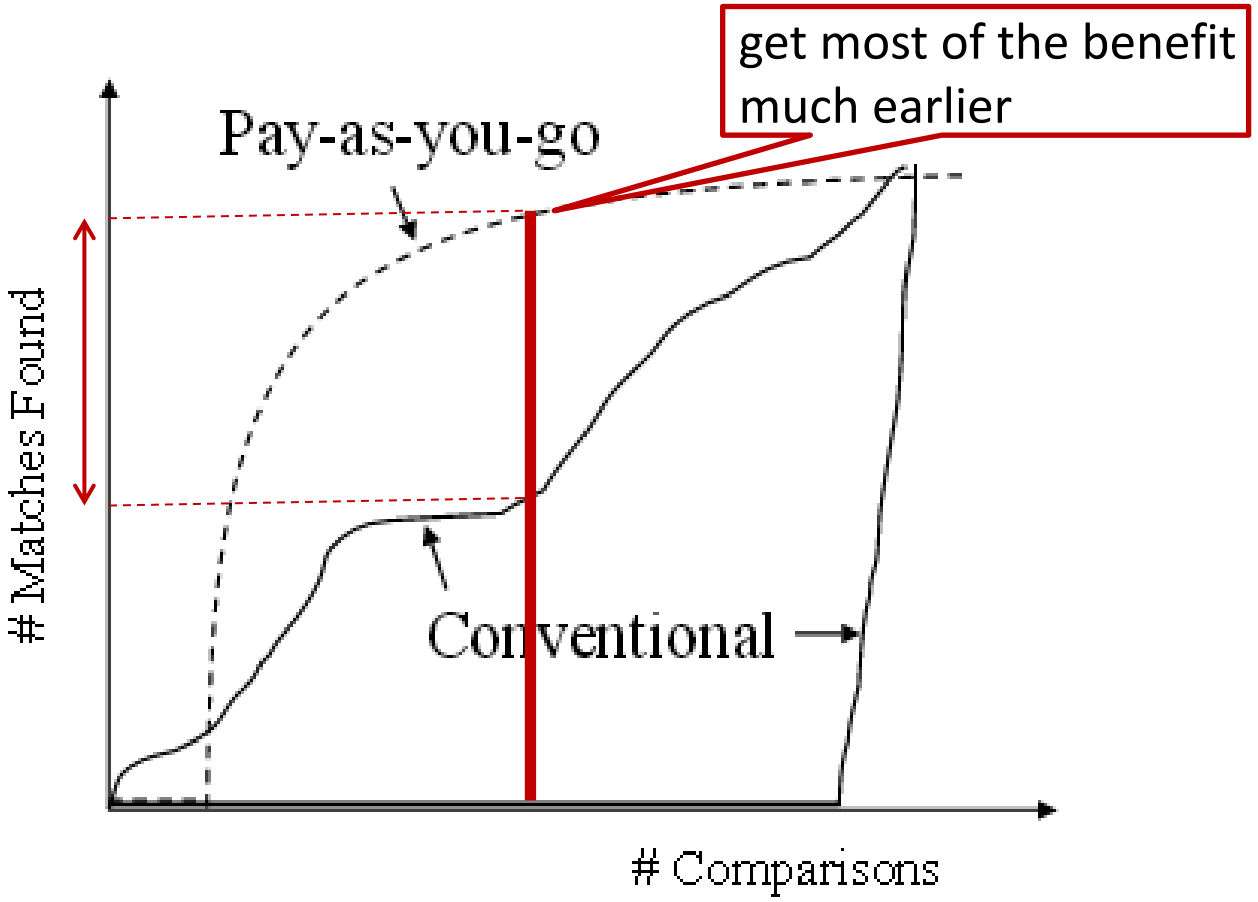
- Progressive, or Pay-as-you-go ER comes is useful



Progressive Blocking

Facts:

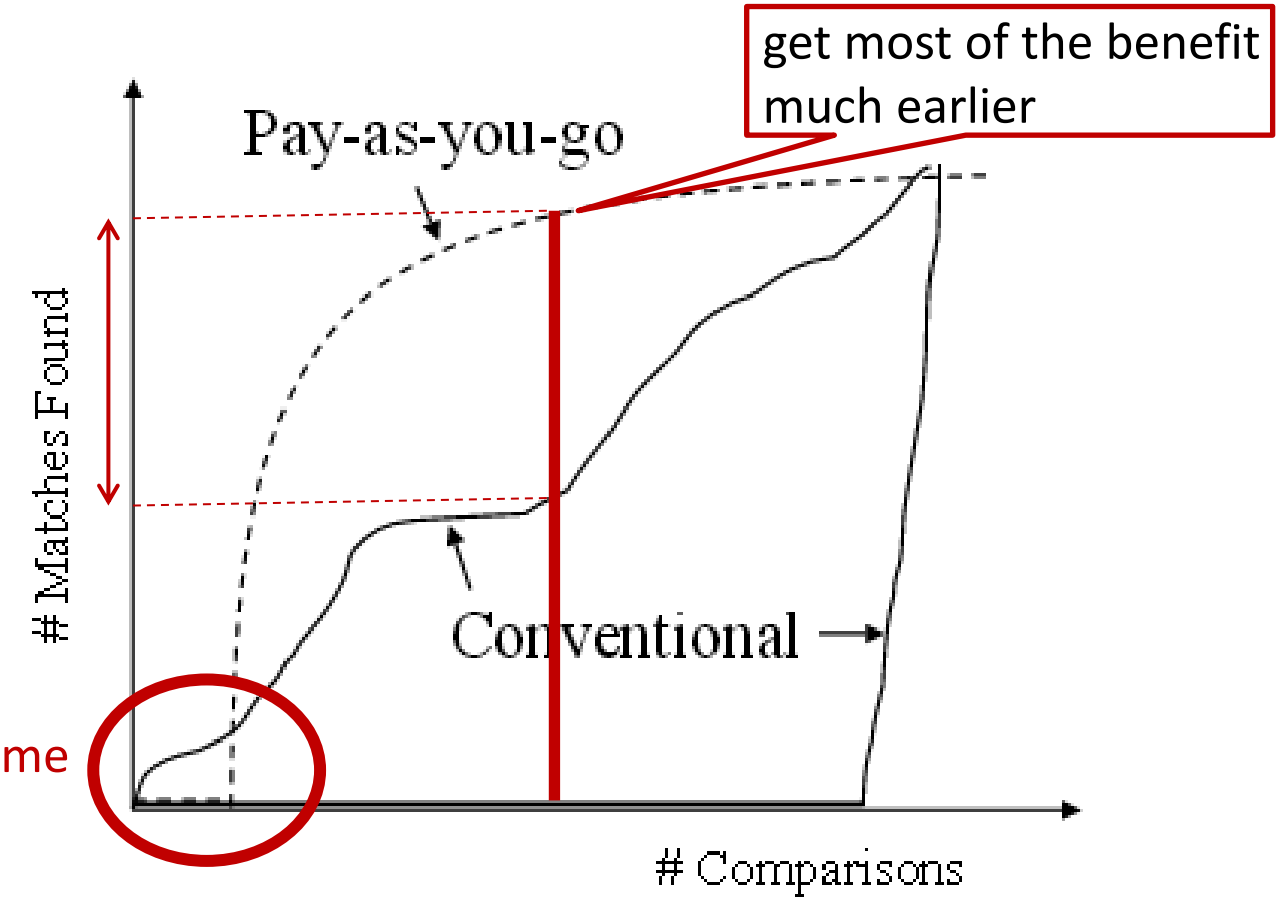
- Progressive, or Pay-as-you-go ER comes is useful



Progressive Blocking

Facts:

- Progressive, or Pay-as-you-go ER comes is useful



may require some pre-processing

Progressive Blocking

Facts:

- Progressive, or Pay-as-you-go ER comes is useful
- Progressive Blocking in its infancy
  - Static methods
    [Whang et. al., TKDE 2013]
  - Dynamic methods
    [Papenbrock et. al., TKDE 2015]
- Only for relational data (schema-aware)

Open Research Directions:

- Schema-agnostic Progressive Blocking

Privacy Preserving Blocking

Facts:

- several applications ask for privacy-preserving ER
- lots of interest in this area [Christen, PADM 2006][Karakasidis et al., 2012][Ziad et al, BTW 2015]

Open Research Directions:

- What is the role of blocking workflow techniques?
  - block building, block filtering, comparison cleaning
- How can existing blocking techniques be adjusted?
- Novel blocking methods for this context

Incremental Blocking

Facts:

- Velocity in Web Data
- Dynamic ER
- Incremental ER [Gruenheid et. al., VLDB 2014]
  - Blocking → black box

Open Research Directions:

- Incremental (Meta-)Blocking

Distributed Blocking

Facts:

- Velocity in Big Data
- Need for even faster/more scalable ER solutions

Open Research Directions:

- What is the best way to use the modern distributed platforms/paradigms?
  - Flink/Spark
- How can we further improve performance of Parallel Meta-blocking?
  - Gelly/Gradoop/GraphX
- Minimize both time performance and total CPU cycles

Part 7:

JedAI Toolkit

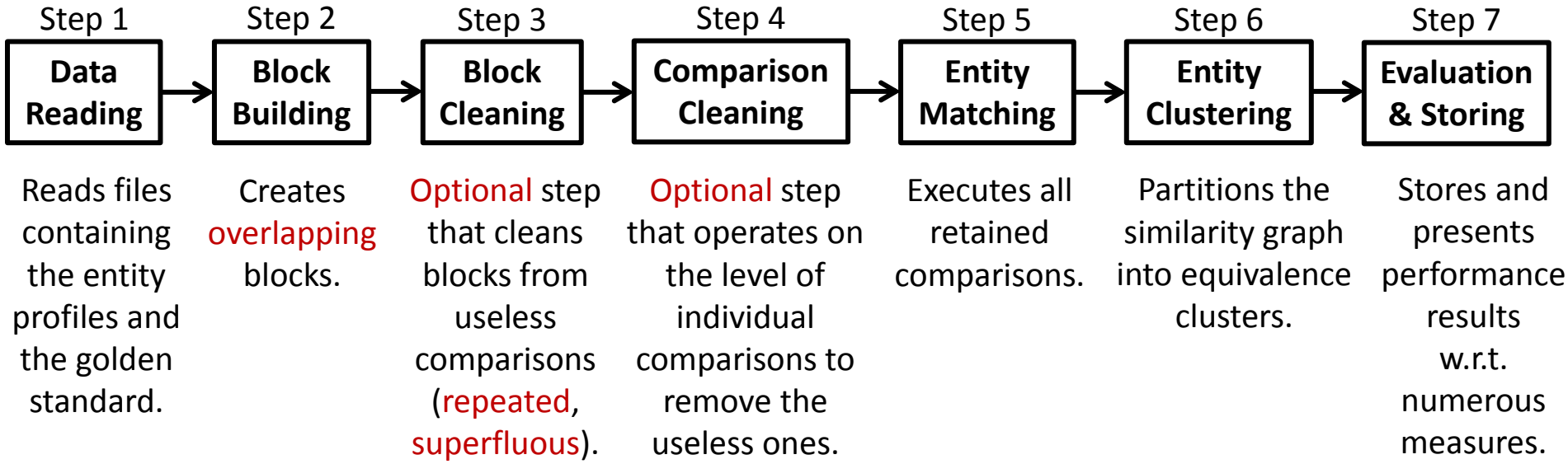
What is the JedAI Toolkit?

JedAI can be used in three ways:

1. As an open source library that implements numerous state-of-the-art methods for all steps of an established end-to-end ER workflow.
2. As a desktop application for ER with an intuitive Graphical User Interface that is suitable for both expert and lay users.
3. As a workbench for comparing all performance aspects of various (configurations of) end-to-end ER workflows.

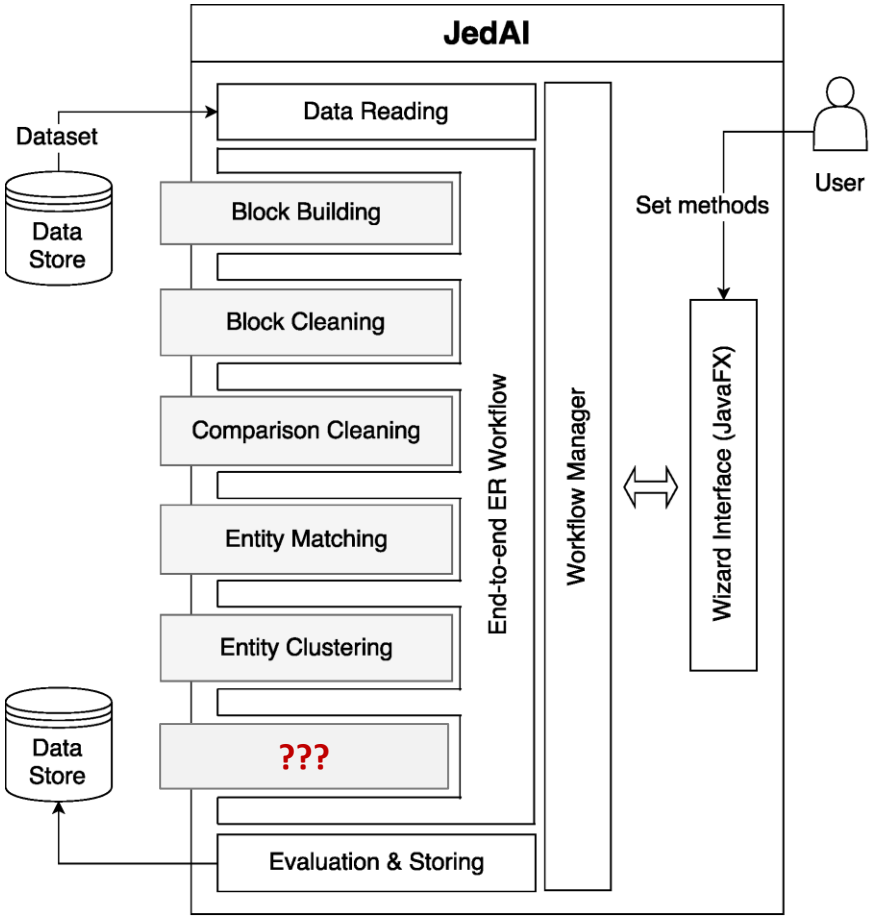
How does the JedAI Toolkit work?

JedAI implements the following schema-agnostic, end-to-end workflow for both Clean-Clean and Dirty ER:



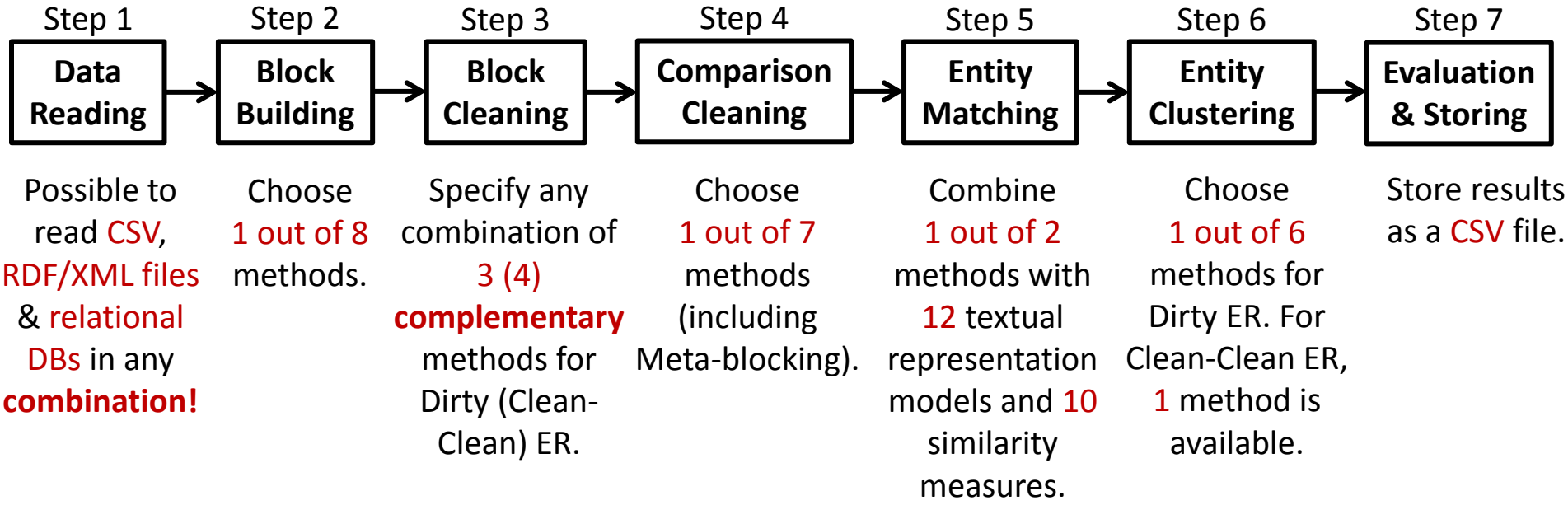
How is the JedAI Toolkit structured?

- Modular architecture: one module per workflow step.
- Extensible architecture (e.g., ontology matching)



How can I build an ER workflow?

JedAI supports several established methods for each workflow step:



Which Blocking Methods are included?

Block Building	Block Cleaning	Comparison Cleaning
Token Blocking	Block Filtering	Comparison Propagation
Sorted Neighborhood	Size-based Block Purging	Cardinality Edge Pruning (CEP)
Extended Sorted Neighborhood	Cardinality-based Block Purging	Cardinality Node Pruning (CNP)
Attribute Clustering	Block Scheduling	Weighted Edge Pruning (WEP)
Q-Grams Blocking		Weighted Node Pruning (WNP)
Extended Q-Grams Blocking		Reciprocal CNP
Suffix Arrays		Reciprocal WNP
Extended Suffix Arrays		

Where can I find JedAI Toolkit?

- Project website: http://jedai.scify.org .
- Github repository of JedAI Library: https://github.com/scify/JedAIToolkit .
- Github repository of JedAI Desktop Application and Workbench: https://github.com/scify/jedai-ui .
  - All code is implemented using Java 8.
  - All code is publicly available under Apache License V2.0.
- Documentation (slides, videos, etc) available at https://github.com/scify/JedAIToolkit/tree/master/documentation .
- When using JedAI, please cite:

  George Papadakis, Leonidas Tsekouras, Emmanouil Thanos, George Giannakopoulos, Themis Palpanas and Manolis Koubarakis: "**JedAI: The Force behind Entity Resolution**", in ESWC 2017.

Which datasets are available for testing?

Several datasets are available for testing at https://github.com/scify/JedAIToolkit .

Can be used for Dirty ER, as well.

Clean-Clean ER (real)	D1 Entities	D2 Entities
Abt-Buy	1,076	1,076
DBLP-ACM	2,616	2,294
DBLP-Scholar	2,516	61,353
Amazon-GP	1,354	3,039
Movies	27,615	23,182
DBPedia	1,190,733	2,164,040

Dirty ER (synthetic)	Entities
10K	10,000
50K	50,000
100K	100,000
200K	200,00
300K	300,00
1M	1,000,000
2M	2,000,000

What are the next steps?

- Version 2.0:
  - Includes support for SPARQL endpoints, multicore functionality and configuration optimization.
  - Available at the end of September, 2017.
- Version 3.0:
  - Includes support for ontology matching, progressive ER as well as a workflow builder.
  - Available at the end of December, 2017.
- Version 4.0:
  - All functionality is implemented in Apache Spark.
  - Available at the end of December, 2018.

Part 8:

Conclusions

Conclusions – Block Building

- Traditional proactive blocking methods only suitable for **relational data**
  - background schema knowledge should be available for their configuration
- Recent lazy blocking methods scale well to heterogeneous, semi-structured **Big Data**
  - **Variety** is addressed with schema-agnostic keys
  - **Volume** is addressed with Block and Comparison Cleaning methods → they trade slightly lower recall, for much higher precision
  - Token Blocking → the only parameter-free blocking method

Conclusions – Block Cleaning

- Coarse-grained functionality:
  - operation at the level of entire blocks
  - low cost (fast) methods
- Only applicable to lazy blocking methods
- They boost the overall performance to a large extent:
  - comparisons drop by orders of magnitude
  - recall drops to a controllable extent (~1-2%)
- Mostly complementary methods
  - multiple Block Cleaning methods can be combined in a single workflow

Conclusions – Comparison Cleaning

- Fine-grained functionality:
  - operate at the level of individual comparisons → computationally intensive process
- Apply to both lazy and proactive methods

- Meta-blocking is the current state-of-the-art
  - Discards both superfluous and redundant comparisons
  - Necessary for reducing comparisons to manageable levels for single-threaded ER workflows
    - **reduces comparisons by orders of magnitude, with recall > 98%**
  - Naturally parallelizable

Big Data Research (BDR) Journal

http://www.journals.elsevier.com/big-data-research/

- New Elsevier journal on topics related to big data
  - advances in big data management/processing
  - interdisciplinary applications
- Editor in Chief for BDR
  - submit your work
  - propose special issues
- google: bdr journal

thank you!
questions?

http://sourceforge.net/projects/erframework

google: themis palpanas
-> publications -> tutorials

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] Mikhail Bilenko, Beena Kamath, Raymond J. Mooney, "Adaptive Blocking: Learning to Scale Up Record Linkage", in ICDM 2006.

[Christen, PADM 2006] Christen P: Privacy-preserving data linkage and geocoding: Current approaches and research directions. PADM held at IEEE ICDM, Hong Kong, 2006.

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

[Efthymiou et. al., BigData 2015] Vasilis Efthymiou, George Papadakis, George Papastefanatos, Kostas Stefanidis, Themis Palpanas, "Parallel meta-blocking: Realizing scalable entity resolution over large, heterogeneous data", in IEEE Big Data 2015.

[Fellegi et. al., JASS 1969] P. Fellegi, A. Sunter, "A theory for record linkage," in Journal of the American Statistical Society, vol. 64, no. 328, 1969.

[Fisher et. al., KDD 2015] Jeffrey Fisher, Peter Christen, Qing Wang, Erhard Rahm, "A Clustering-Based Framework to Control Block Sizes for Entity Resolution" in KDD 2015.

[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.

[Gruenheid et. al., VLDB 2014] Anja Gruenheid, Xin Luna Dong, Divesh Srivastava, "Incremental Record Linkage", in PVLDB 2014.

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

References – Part B

[Karakasidis et al., SAC 2012] Karakasidis A and Verykios VS: Reference table based k-anonymous private blocking. Symposium on Applied Computing, 2012.

[Kenig et. al., IS 2013] Batya Kenig, Avigdor Gal, "MFIBlocks: An effective blocking algorithm for entity resolution", in Inf. Syst. 2013.

[Ma et. Al., WSDM 2013] Y. Ma, T. Tran, "TYPiMatch: type-specific unsupervised learning of keys and key values for heterogeneous web data integration", in WSDM 2013.

[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] Matthew Michelson, Craig A. Knoblock, "Learning Blocking Schemes for Record Linkage", in AAAI 2006.

[Papadakis et. al., EDBT 2016] George Papadakis, George Papastefanatos, Themis Palpanas, Manolis Koubarakis, "Scaling Entity Resolution to Large, Heterogeneous Data with Enhanced Meta-blocking", in EDBT 2016.

[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., 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., TKDE 2013] George Papadakis, Ekaterini Ioannou, Themis Palpanas, Claudia Niederee, Wolfgang Nejdl, "A Blocking Framework for Entity Resolution in Highly Heterogeneous Information Spaces", in IEEE TKDE 2013.

References – Part C

[Papadakis et. al., TKDE 2014] George Papadakis, Georgia Koutrika, Themis Palpanas, Wolfgang Nejdl, "Meta-Blocking: Taking Entity Resolution to the Next Level", in IEEE TKDE 2014.

[Papadakis et. al., VLDB 2014] G. Papadakis, G. Papastefanatos, G. Koutrika, "Supervised Meta-blocking", in PVLDB 2014.

[Papadakis et. al., VLDB 2015] George Papadakis, George Alexiou, George Papastefanatos, Georgia Koutrika, "Schema-agnostic vs Schema-based Configurations for Blocking Methods on Homogeneous Data", in PVLDB 2015.

[Papadakis et. al., VLDB 2016] George Papadakis, Jonathan Svirsky, Avigdor Gal, Themis Palpanas, "Comparative Analysis of Approximate Blocking Techniques for Entity Resolution", in PVLDB 2016.

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

[Papenbrock et. al., TKDE 2015] Thorsten Papenbrock, Arvid Heise, Felix Naumann, "Progressive Duplicate Detection", in IEEE TKDE 2015.

[Sarma et. al, CIKM 2012] Anish Das Sarma, Ankur Jain, Ashwin Machanavajjhala, Philip Bohannon, "An automatic blocking mechanism for large-scale de-duplication tasks" in CIKM 2012.

[Simonini et. al, VLDB 2017] Giovanni Simonini, Sonia Bergamaschi and H.V. Jagadish, "Blast: a Loosely schema-aware Meta-blocking Approach for Entity Resolution" in VLDB 2017.

[Whang et. Al, SIGMOD 2009] Whang, D. Menestrina, G. Koutrika, M. Theobald, H. Garcia-Molina, "Entity resolution with iterative blocking", in SIGMOD 2009.

[Whang et. al., TKDE 2013] Steven Euijong Whang, David Marmaros, Hector Garcia-Molina, "Pay-As-You-Go Entity Resolution", in IEEE TKDE 2013.

[Ziad et al, BTW 2015] Ziad Sehili, Lars Kolb, Christian Borgs, Rainer Schnell, Erhard Rahm: Privacy Preserving Record Linkage with PPJoin. BTW 2015.