Blocking Techniques for Web-scale Entity Resolution

George Papadakis – Themis Palpanas
IMIS, Athena RC
Paris Descartes University
gpapadis@imis.athena.innovation.gr
themis@mi.parisdescartes.fr
Outline

1. Introduction to Entity Resolution
2. Introduction to Blocking
3. Blocking Methods for Databases
4. Blocking Methods for Web Data
5. Meta-blocking
6. Block Processing Techniques
7. ER framework
Part 1:
Introduction to Entity Resolution
Entities: an invaluable asset

“Entities” is what a large part of our knowledge is about:

- Persons
- Organizations
- Products
- Events
- Projects
- Locations

Papadakis & Palpanas, Tutorial@WISE14, 12. October 2014
However ...

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

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, ...

Papadakis & Palpanas, Tutorial@WISE14, 12. October 2014
However ...

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

- London
- 런던
- لندن
- لندن
- لندن
- لندن
- لندن
- ロンドン
- лондон
- இலண்டன்
- ლონდონი
- Llundain
- Londain
- Londe
- Londen
- Londinium
- London
- Londona
- Londonas
- Londoni
- Londono
- Londra
- Londres
- Londrez
- Londyn
- Lontoo
- Loundres
- Luân
- Đôn
- Lunden
- Lundúnir
- Lunnainn
- Lunnnon
- لندن
- Londinium
- Larry
- Lounes
- لندن
- لندن
- لندن
- لوندون
- لندن
- לונדון
- Λονδίνο
- Лёндан
- Лондан
- Лондон
- Лондон
- Лондон
- Лондон
- لندن

- 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/

Papadakis & Palpanas, Tutorial@WISE14, 12. October 2014
... 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
- ...
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”? 

- News about London
- Reviews on hotels in London
- Wiki pages about the London
- Pictures and tags about London
- Social networks in London
- Videos and tags for London

Papadakis & Palpanas, Tutorial@WISE14, 12. October 2014
Preliminaries on Entity Resolution

**Entity Resolution** [Christen, TKDE2011]:
identifies and aggregates the different entity profiles/records that actually describe the same real-world object.

Application areas:
Linked Data, Social Networks, census data, price comparison portals

Useful because:
• improves data quality and integrity
• fosters re-use of existing data sources.
Types of Entity Resolution

The input of ER consists of entity collections that can be of two types [Christen, TKDE2011]:

- **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

(Papadakis & Palpanas, Tutorial@WISE14, 12. October 2014)
Types of Entity Resolution

The input of ER consists of entity collections that can be of two types [Christen, TKDE2011]:

• **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

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

• **Dirty-Dirty ER**

Papadakis & Palpanas, Tutorial@WISE14, 12. October 2014
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 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 to large entity collections (e.g., Web Data)

Solution: **Blocking**
- group similar entities into blocks
- execute comparisons only inside blocks
- approximate solution
Computational cost

- Blocking
- Pairs of Duplicates

Brute-force approach
Part 2:

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.
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:

- Pairs Completeness:  \( PC = \frac{\text{detected matches}}{\text{existing matches}} \)  
  (recall)

- Pairs Quality:  \( PQ = \frac{\text{detected matches}}{\text{executed comparisons}} \)  
  (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 $PC$ and $PQ$ are maximized.

**disclaimer:**

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

1. Definition of blocking keys
   – Supervised
   – Unsupervised

2. Dependency on schema
   – Schema-based
   – Schema-agnostic

3. Redundancy
   – Disjoint blocks
   – Overlapping blocks
     ▪ Redundancy-positive
     ▪ Redundancy-neutral
     ▪ Redundancy-negative
# Unsupervised Blocking Methods

<table>
<thead>
<tr>
<th>Disjoint Blocks</th>
<th>Overlapping Blocks</th>
<th>Overlapping Blocks</th>
</tr>
</thead>
</table>
Part 3:

Blocking Methods for Databases
General Principles

Mostly schema-based techniques.
Rely on two assumptions:
1. A-priori known schema $\rightarrow$ 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
Standard Blocking

Earliest, simplest form of blocking.

Algorithm:
1. Select the most appropriate attribute name w.r.t. noise and distinctiveness.
2. Transform every value into a single 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!*
Example of Standard Blocking

Blocks on zip_code:
Q-grams Blocking [Baxter et. al., KDD 2003] [Gravano et. al., VLDB 2001]

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 $\rightarrow$ higher computational cost
Suffix Array Blocking [Aizawa et. al., WIRI 2005][de Vries et. al., CIKM 2009]

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

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

- **Advantage:**
  robust to noisy BKVs

- **Drawback:**
  larger blocks $\rightarrow$ higher computational cost

Papadakis & Palpanas, Tutorial@WISE14, 12. October 2014
Sorted Neighborhood [Hernandez et. al., SIGMOD 1995]

1. Entities are sorted in alphabetic order of BKs.
2. A window of fixed size slides over the sorted list.
3. At each iteration, it compares the entities that co-occur within the window.
Sorted Neighborhood [Hernandez et. al., SIGMOD 1995]

1. Entities are sorted in alphabetic order of BKs.
2. A window of fixed size slides over the sorted list.
3. At each iteration, it compares the entities that co-occur within the window.
Sorted Neighborhood [Hernandez et. al., SIGMOD 1995]

1. Entities are sorted in alphabetic order of BKs.
2. A window of fixed size slides over the sorted list.
3. At each iteration, it compares the entities that co-occur within the window.
Sorted Neighborhood  [Hernandez et. al., SIGMOD 1995]

1. Entities are sorted in alphabetic order of BKs.
2. A window of fixed size slides over the sorted list.
3. At each iteration, it compares the entities that co-occur within the window.
Canopy Clustering [McCallum et. al., KDD 2000]

Points at a distance < T2 cannot be canopy centers themselves and belong to the canopy centered at Point P.

Points at a distance > T1 are considered too far away do not belong to the canopy.

Points at a distance > T2 but less than T2 from the center point are a part of the canopy but can also be canopy centers themselves.

Center of the Canopy(P)
Summary of Blocking for Databases [Christen, TKDE2011]

They typically employ **redundancy** to ensure robustness in the context of noise at the cost of lower efficiency.

Drawbacks:

1. Too many parameters to be configured
   
   Canopy Clustering has the following parameters:
   
   I. String matching method
   
   II. Threshold $t_1$
   
   III. Threshold $t_2$

2. Schema-dependent
Part 4:

Blocking Methods for Web Data
Characteristics of Web Data

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 for 10,000 entity types
- BTC09: 136K attribute names

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

Papadakis & Palpanas, Tutorial@WISE14, 12. October 2014
Token Blocking  [Papadakis et al., WSDM2011]

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

Attribute-agnostic blocking scheme:
• completely ignores attribute names
• considers all attribute values
• redundancy-positive blocks
• parameter-free!

*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
- hdq=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 inside each cluster, without affecting effectiveness → smaller blocks, higher efficiency.

Algorithm:

• Create a graph with a node for every attribute name
• For each attribute name \( n_i \)
  − Find the most similar \( n_j \)
  − If \( \text{sim}(n_i,n_j) > 0 \), add an edge \( <n_i,n_j> \)
• Extract connected components
• Put all singleton nodes in a “glue” cluster
Attribute-Clustering Blocking [Papadakis et. al., TKDE2013]

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

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. Unlike Schema Matching, it scales to the extreme levels of heterogeneity of Web Data.
Evidence for Semantic Web Blocking

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

1. **Infix** [Papadakis et al., iiWAS 2010]

![Infix Example](http://dblp.13s.de/d2r/resource/publications/books/sp/wooldridgeV99
http://bibsonomy.org/uri/bibtexkey/books/sp/wooldridgeV99
Infix
/ThalmannN99
/ThalmannN99
/dblp)

2. **Infix Profile**

3. **Literal Profile**

![Literal Profile Example](http://dbpedia.org/resource/Barack_Obama)
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>

Papadakis & Palpanas, Tutorial@WISE14, 12. October 2014
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 for higher robustness and effectiveness.
Summary of Blocking for Web Data

attribute-agnostic functionality → no schema semantics so as to handle any level of heterogeneity

redundancy to reduce the likelihood of missed matches → high recall

redundancy-positive blocks

Drawbacks:

• the blocks are overlapping (i.e., repeated comparisons)
• high number of comparisons between irrelevant entities → low precision
Part 5:
Meta-blocking
Meta-blocking [Papadakis et. al., TKDE]

Goal:
restructure a redundancy-positive block collection into a new one that contains a substantially lower number of comparisons, while being equally effective ($\Delta PC \approx 0$, $\Delta PQ \gg 0$).
Type of pair-wise comparisons

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

1. **Matching** if $p_i \equiv p_j$.
2. **Redundant** if $p_i$ and $p_j$ co-occur and will be compared in another block.
3. **Superfluous** if $p_i$ or $p_j$ or both of them have been matched to some other entity (Clean-Clean ER).
4. **Non-matching** if $p_i \neq p_j$ and the comparison is not redundant (for Dirty ER). For Clean-Clean ER, it should not be superfluous either.
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
- hdq=California
- status=active

**Entity 4**
- firstName=Ann
- lastName=Veneman
- residence=California

Papadakis & Palpanas, Tutorial@WISE14, 12. October 2014
Meta-blocking [Papadakis et. al., TKDE]

Goal:

restructure a redundancy-positive block collection into a new one that contains substantially lower number of redundant and non-matching comparisons, while maintaining the original number of matching ones ($\Delta PC \approx 0$, $\Delta PQ > 0$).
Meta-blocking [Papadakis et. al., TKDE]

Goal:

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

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

B → Graph Building → $G_B$ → Edge Weighting → $G_B^w$ → Graph Pruning → $G_B^p$ → Block Collecting → B'

Papadakis & Palpanas, Tutorial@WISE14, 12. October 2014
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 or bit arrays.

Papadakis & Palpanas, Tutorial@WISE14, 12. October 2014
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

Papadakis & Palpanas, Tutorial@WISE14, 12. October 2014
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: $K = BPE \cdot |E|/2$

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, $k = BPE - 1$
Block Collecting

Transform the pruned blocking graph into a new 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
Part 6:

Block Processing Techniques
General Principles

Goals:

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

without affecting matching comparisons (i.e., effectiveness).
General Principles

Goals:

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

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

Taxonomy of techniques:

<table>
<thead>
<tr>
<th>Granularity</th>
<th>Comparison’s Type</th>
<th>Repeat Method</th>
<th>Superfluity Method</th>
<th>Non-match method</th>
<th>Scheduling method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block-refinement</td>
<td>-</td>
<td>-</td>
<td>1. Block Purging</td>
<td>Block Scheduling</td>
<td></td>
</tr>
<tr>
<td>Comparison-refinement</td>
<td>Comparison Propagation</td>
<td>Duplicate Propagation</td>
<td>Comparison Pruning</td>
<td>Comparison Scheduling</td>
<td></td>
</tr>
</tbody>
</table>

Papadakis & Palpanas, Tutorial@WISE14, 12. October 2014
**Block Purging** [Papadakis et al., WSDM2011] & [Papadakis et al., WSDM2012]

**Oversized blocks:** many, unnecessary comparisons (redundant, non-matching, superfluous).

**Block Purging:** discards oversized blocks 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.

Papadakis & Palpanas, Tutorial@WISE14, 12. October 2014
Comparison Propagation [Papadakis et al., SWIM 2011]

• Eliminates all redundant comparisons at no cost in recall → naïve approach does not scale
• Enumerates Blocks
• Least Common Block Index condition.
Part 7:

ER Framework
ER-Framework

- Offers a suite of blocking methods for benchmarking.
- Continuous updates.
- Plan to add GUI, documentation and more methods by the end of 2015.
- Established real-world and synthetic datasets available.
Structure of the ER-Framework

- Effectiveness Layer
  - Disk-based Methods
  - Memory-based Methods
- Efficiency Layer
  - Block-refinement
  - Comparison-refinement
  - Meta-blocking
- Utilities, Data Structures,...
Effectiveness Layer

• Common interface for all methods imposed by AbstractBlockingMethod.
  – Input: dataset 1, dataset 2 (null for Dirty ER) in the form of List<EntityProfile> and parameters, depending on the approach
  – Output: block collection of the form List<AbstractBlock> returned by buildBlocks().

  • It contains objects of type UnilateralBlock for Dirty ER and of type BilateralBlock for Clean-Clean ER.

• Disk-based methods: first store blocks as a Lucene index on a specified directory.
Efficiency Layer

Common interface for all methods imposed by AbstractEfficiencyMethod.

- Input: a block collection of the form `List<AbstractBlock>`. 
- Output: changes to the elements of the input block collection.
- Functionality implemented by `applyProcessing()`.
Measuring Performance

Ground-truth of the form $\text{Set<IdDuplicates>}$, where $\text{IdDuplicates}$ contains a pair of entity ids.

Class $\text{BlockStatistics}$ measures the performance of a block collection wrt:

- PC, PQ, $||B||$, $|D_B|$, BC, CC.
Thank You!
References – Part A


References – Part B


References – Part C


