Blocking for Large-Scale Entity Resolution
Challenges, Algorithms, Practical Examples

George Papadakis – Themis Palpanas
MaDgIK, University Athens
gpapadis@di.uoa.gr
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”? 

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
However ...

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

London 런던 लंडन لندن Londýn Лондон Лондон Londain Londe Londen Londen Londinium London Londona Londonas Londoni Londono Londra Londres Londrez Londyn Lontoo Loundres Luân Đôn Lunden Lundúnir Lunnainn Lunnon لندن لندن لندن لوندون Λονδίνο Λένδαν Λόνδαν Λόνδον Λόνδον لندن Lndon 伦敦 ...
However ... 

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

London 런던 लंडनLANDON ロンドン لندن لندن لندن لندن لندن Llundain Londain Londe Londen Londen Loden Londinium London Londona Londonas Londoni Londono Londra Londres Londrez Londyn Lontoo Loundres Luân Đôn Lunden Lundúnir Lunnainn Lunnon لندن لندن لندن لندن Louðívo Лёндан Лондан Лондон Лондон Londen لندن لندن لندن 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, ...
However ...

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

London 런던 लंडन ロンドン লন্ডন Llundain Londain Londe Londen Londen Londen Londinium London Londona Londonas Londoni Londono Londra Londres Londrez Londyn Loundres Luàn Đọn Lunden Lundúnir Lunnainn Lunnon لندن لندن لندن لوندون Λονδίνο Лёндан Лондан Lондон Londen Londen Londres... 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://dbpedia.org/resource/Category:London ...

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
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
- ...

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
... 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”?

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

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
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

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
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

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**
- **Dirty-Dirty ER** (a.k.a. *Deduplication* in databases)

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
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!)
Input: Entity Collection $E$

Computational cost

| $|E|$ entities |
|----------------|

- Brute-force approach
- Blocking
- Duplicate Pairs
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: 92%
Total Running time: **5 hours**

Papadakis & Palpanas, Tutorial@ICDE16,
May 18th, 2016
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. ER framework
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 → 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: \( PC = \frac{\text{detected matches}}{\text{existing matches}} \) (optimistic recall)

- Pairs Quality: \( PQ = \frac{\text{detected matches}}{\text{executed comparisons}} \) (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.

cautions:

• Emphasis on Pairs Completeness (PC).
  – if two entities are matching then they should coincide at some block
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, \( s \equiv t \) if:
     
     JaccardSimilarity(\( s.\text{title} \), \( t.\text{title} \)) \( > 0.80 \) AND
     EditDistance(\( s.\text{venue} \), \( t.\text{venue} \)) \( < 3 \)

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

- **Block Building**
  - Lazy blocking methods

- **Block Cleaning**
  - Block-refinement methods

- **Comparison Cleaning**
  - Comparison-refinement methods

---

Proactive blocking methods
# Blocks- and Signature-wise Categorization of Block Building Methods

<table>
<thead>
<tr>
<th>Disjoint Blocks</th>
<th>Overlapping Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema-based</strong></td>
<td></td>
</tr>
<tr>
<td>Standard Blocking</td>
<td>(Extended) Canopy Clustering</td>
</tr>
<tr>
<td></td>
<td>1. (Extended) Sorted Neighborhood</td>
</tr>
<tr>
<td></td>
<td>2. MFIBlocks</td>
</tr>
<tr>
<td></td>
<td>1. (Extended) Q-grams Blocking</td>
</tr>
<tr>
<td></td>
<td>2. (Extended) Suffix Arrays</td>
</tr>
<tr>
<td><strong>Schema-agnostic</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>1. Token Blocking</td>
</tr>
<tr>
<td></td>
<td>2. Agnostic Clustering</td>
</tr>
<tr>
<td></td>
<td>3. TYPiMatch</td>
</tr>
<tr>
<td></td>
<td>4. URI Semantics Blocking</td>
</tr>
</tbody>
</table>
Block Processing Methods
[Papadakis et. al., VLDB 2016]

Mostly for redundancy-positive block building methods.

Blocking 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 3:

Block Building for Relational Data
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
Overview of Schema-based Methods

- **Standard Blocking**
  - Sorted Neighborhood
    - Extended Sorted Neighborhood
  - Suffix Arrays
    - Extended Suffix Arrays
- **Q-grams Blocking**
  - Extended Q-grams Blocking
    - Extended Canopy Clustering
      - Extended Canopy Clustering
  - MFIBlocks
- **Extended Q-grams Blocking**

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
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

- Standard Blocking
  - Sorted Neighborhood
    - Extended Sorted Neighborhood
  - Q-grams Blocking
    - Extended Q-grams Blocking
    - MFIBlocks
  - Suffix Arrays
    - Extended Suffix Arrays
- Canopy Clustering
  - Extended Canopy Clustering
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.
3. At each iteration, it compares the entities that co-occur within the window.

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
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.
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.
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.
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.
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

- Standard Blocking
  - Sorted Neighborhood
    - Extended Sorted Neighborhood
      - Q-grams Blocking
      - MFIBlocks
      - Extended Q-grams Blocking
  - Suffix Arrays
    - Extended Suffix Arrays
      - Canopy Clustering
    - Extended Canopy Clustering

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
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 $\rightarrow$ 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: $l_{\text{min}} = \max(1, \lfloor k \cdot t \rfloor)$, where $k$ is the number of $q$-grams from the original BK.

Example:

for BK= 91456, $q=2$ and $t=0.9$, we have $l_{\text{min}}=3$ 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
• Usually the most effective blocking method for relational data $\rightarrow$ maximizes PQ

Cons:
• Difficult to configure
• Time consuming
Overview of Schema-based Methods

Standard Blocking

Sorted Neighborhood

Extended Sorted Neighborhood

Q-grams Blocking

Extended Q-grams Blocking

MFIBlocks

Suffix Arrays

Extended Suffix Arrays

Canopy Clustering

Extended Canopy Clustering

Papadakis & Palpanas, Tutorial@ICDE16,
May 18th, 2016
Canopy Clustering [McCallum et. al., KDD 2000]

Blocks on similarity of BKs.
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

- Standard Blocking
  - Sorted Neighborhood
    - Extended Sorted Neighborhood
  - Q-grams Blocking
    - Extended Q-grams Blocking
  - MFIBlocks
    - Extended Canopy Clustering
- Suffix Arrays
  - Extended Suffix Arrays
Suffix Arrays Blocking [Aizawa et. al., WIRI 2005]

Blocks on the equality of BKs.

Converts every BKV 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 $b_M$, i.e., 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 $l_{\text{min}}$ characters.

For $l_{\text{min}} = 3$, 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 $\rightarrow$ 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 methods →
  very high, robust recall at the cost of more comparisons
• For proactive methods →
  relative recall gets higher for more comparisons,
  absolute recall depends on block constraints
Part 4:

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 insert not only attribute values but also 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: 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

Loose Schema Binding

DATASET 2

Entity 3
- organization: unicef
- California
- status: active
- Los Angeles, 91335

Entity 4
- firstName: Ann
- lastName: Veneman
- residence: California
- zip_code: 90210

Split values

Attribute Heterogeneity

Noise

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
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 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

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
- hq=California
- status=active

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

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
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 $\rightarrow$ smaller blocks, higher efficiency.
Attribute-Clustering Functionality

Algorithm

• Create a graph, where every node corresponds to an attribute name and aggregates its attribute values
• For each attribute name/node $n_i$
  – Find the most similar node $n_j$
  – 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

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.
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

Algorithm:
1. Create a directed graph $G$, where nodes correspond to tokens and edges connect those co-occurring in the same entity profile and are weighted according to conditional co-occurrence probability.
2. Convert $G$ to undirected graph $G'$ and get maximal cliques (parameter $\theta$).
3. Create an undirected graph $G''$, where nodes correspond to cliques and edges connect the frequently co-occurring cliques (parameter $\varepsilon$).
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

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

High Recall in the context of noise entity profiles and extreme schema heterogeneity because:
1. redundancy to reduce the likelihood of missed matches.
2. attribute-agnostic functionality that requires no schema semantics.

Low Precision because:
• the blocks are overlapping \(\rightarrow\) redundant comparisons
• high number of comparisons between irrelevant entities \(\rightarrow\) superfluous comparisons

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
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

Redundant Comparison

Superfluous Comparison
Part 5:

Block Processing Techniques
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.

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
Distributions of Block Sizes and Duplicates

Number of Blocks vs. Block Cardinality

% of Duplicates vs. Block Cardinality

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
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

Entity Index

- **Entity 1** → \{ unicef, California \}
- **Entity 2** → \{ Ann, Veneman, unicef, California \}
- **Entity 3** → \{ unicef, California \}
- **Entity 4** → \{ Ann, Veneman, California \}

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
Block Clustering [Fisher et. al., KDD 2015]

Main idea:
• restrict the size of every block into $[b_{\text{min}}, b_{\text{max}}]$  
  – necessary in applications like privacy-preserving ER  
  – $|\|B\||$ increases linearly with $|E|$  

Algorithm
• recursive agglomerative clustering  
  – merge similar blocks with size lower than $b_{\text{min}}$  
  – split blocks with size larger than $b_{\text{max}}$  
• 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. Builds Entity Index
  2. Least Common Block Index condition.

![Diagram](image)
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 \( p'_i = p'_j = \langle p_i, p_j \rangle \) and update them in all associated blocks
      – Place in Q all associated blocks that are not already in it
Part 6:

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

**Optimized Block Building + Block Cleaning**

Overhead time: <30 mins
Comparisons: \(4.96 \cdot 10^{10}\) (on average across all lazy methods)
Recall: 99%
Total Running time: 26 days !!
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 ($\Delta PC \approx 0$, $\Delta PQ \gg 0$)
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 ($\Delta PC \approx 0$, $\Delta PQ \gg 0$)

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

1. Graph Building
2. Edge Weighting
3. Graph Pruning
4. Block Collecting

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
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!

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
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.

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
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
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, \( k=BPE-1 \)
Meta-blocking Challenges

1. Time Efficiency
   - Bottleneck: edge weighting
   - Depends on $|B|$, BPE
     - $|E| = 3.4 \times 10^6$, $|B| = 4 \times 10^{10}$, BPE=15 $\rightarrow$ 3 hours
     - $|E| = 7.4 \times 10^6$, $|B| = 2 \times 10^{11}$, BPE=40 $\rightarrow$ 186 hours

2. Effectiveness
   Simple pruning rules
Enhancing Meta-blocking Efficiency

• Block Filtering
  – $r=0.8 \rightarrow$ 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)
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

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
Pre-processing (advanced)

Key Value
<table>
<thead>
<tr>
<th>e₁</th>
<th>b₁,b₄,b₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>e₂</td>
<td>b₁</td>
</tr>
<tr>
<td>e₃</td>
<td>b₁,b₄</td>
</tr>
<tr>
<td>e₄</td>
<td>b₄,b₅</td>
</tr>
</tbody>
</table>

Map → Group by key → Reduce

Key Value
<table>
<thead>
<tr>
<th>b₁</th>
<th>[e₁,b₁,b₄,b₆]</th>
</tr>
</thead>
<tbody>
<tr>
<td>b₄</td>
<td>[e₁,b₁,b₄,b₆]</td>
</tr>
<tr>
<td>b₆</td>
<td>[e₁,b₁,b₄,b₆]</td>
</tr>
<tr>
<td>b₁</td>
<td>[e₂,b₁]</td>
</tr>
</tbody>
</table>

Map → Group by key → Reduce

Key Value
<table>
<thead>
<tr>
<th>b₁</th>
<th>[e₁,b₁,b₄,b₆]</th>
</tr>
</thead>
<tbody>
<tr>
<td>b₁</td>
<td>[e₂,b₁]</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Reduce

Key Value
| b₁  | [e₁,b₁,b₄,b₆], [e₂,b₁], [e₃,b₁,b₄] |
|     |                                           |
| ... |                                           |

Reduce

Key Value
| b₄  | [e₁,b₁,b₄,b₆], [e₃,b₁,b₄], [e₄,b₄,b₅] |
|     |                                           |
| ... |                                           |

Reduce

Key Value
| b₅  | [e₄,b₄,b₅] |
|     |            |
| ... |            |

Reduce

Key Value
| b₆  | [e₁,b₁,b₄,b₆] |
|     |               |
| ... |               |

Reduce
Meta-blocking (advanced) – WEP & JS

Key | Value
---|---
$b_1$ | $[e_1, b_1, b_4, b_6]$, $[e_2, b_1]$, $[e_3, b_1, b_4]$  
... | ...  
b_4 | $[e_1, b_1, b_4, b_6]$, $[e_3, b_1, b_4]$, $[e_4, b_4, b_5]$  
... | ...

Map

Key | Value
---|---
e_1.e_2 | 1/3  
e_1.e_3 | 2/3  
e_2.e_3 | 1/2  
... | ...  

Group by key

Key | Value
---|---
e_1.e_2 | 1/3  
... | ...  
e_1.e_3 | 2/3  
... | ...

Count #keys

Sum up values

Reduce

Key | Value
---|---
e_1.e_3 | 2/3  
... | ...

Reduce

Key | Value
---|---
e_2.e_3 | 1/2  
... | ...

Reduce

... | ...

... | ...

... | ...
Enhancing Meta-blocking Effectiveness

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

**Goal:**
more accurate and comprehensive rules 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
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.
Part 7:

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

Vision:
• Plug-and-play blocking
• Data-driven configuration
Progressive Blocking

Facts:

• Progressive or Pay-as-you-go ER comes in handy
• Progressive Blocking in its infancy
  – Static methods
    [Whang et. al., TKDE 2013]
  – Dynamic methods
    [Papenbrock et. al., TKDE 2015]
• Only for relational data

Vision:
• Schema-agnostic Progressive Blocking
Incremental Blocking

Facts:
• Velocity in Web Data
• Dynamic ER
• Incremental Record Linkage [Gruenheid et. al., VLDB 2014]
  – Blocking → black box

Vision:
• Incremental (Meta-)Blocking
ER-Framework

• Offers a suite of blocking methods for benchmarking.
• Code in Java 8 (Netbeans project) available at: http://sourceforge.net/projects/erframework . Tutorial slides also available for download from this site.
• Continuous updates.
• Work in progress: GUI & documentation!

Papadakis & Palpanas, Tutorial@ICDE16,
May 18th, 2016
## Implemented Methods

<table>
<thead>
<tr>
<th>Block Building</th>
<th>Block-refinement</th>
<th>Comparison-refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token Blocking</td>
<td>Block Filtering</td>
<td>Comparison Propagation</td>
</tr>
<tr>
<td>Attribute Clustering</td>
<td>Block Pruning</td>
<td>Comparison Pruning</td>
</tr>
<tr>
<td>Canopy Clustering</td>
<td>Block Scheduling</td>
<td>Comparison Scheduling</td>
</tr>
<tr>
<td>Extended Canopy Clustering</td>
<td>Size-based Block Purging</td>
<td>Iterative Blocking</td>
</tr>
<tr>
<td>Q-Grams Blocking</td>
<td>Cardinality-based Block Purging</td>
<td>Meta-blocking*</td>
</tr>
<tr>
<td>Extended Q-Grams Blocking</td>
<td></td>
<td>Supervised Meta-blocking*</td>
</tr>
<tr>
<td>Sorted Neighborhood</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extended Sorted Neighborhood</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suffix Arrays</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extended Suffix Arrays</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYPiMatch</td>
<td></td>
<td>* all algorithms</td>
</tr>
</tbody>
</table>

Papadakis & Palpanas, Tutorial@ICDE16, May 18th, 2016
### Available Benchmark Datasets

<table>
<thead>
<tr>
<th>Clean-Clean ER (real)</th>
<th>D1 Entities</th>
<th>D2 Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abt-By</td>
<td>1,076</td>
<td>1,076</td>
</tr>
<tr>
<td>DBLP-ACM</td>
<td>2,616</td>
<td>2,294</td>
</tr>
<tr>
<td>DBLP-Scholar</td>
<td>2,516</td>
<td>61,353</td>
</tr>
<tr>
<td>Amazon-GP</td>
<td>1,354</td>
<td>3,039</td>
</tr>
<tr>
<td>Movies</td>
<td>27,615</td>
<td>23,182</td>
</tr>
<tr>
<td>DBPedia</td>
<td>1,190,733</td>
<td>2,164,040</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dirty ER (synthetic)</th>
<th>Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K</td>
<td>10,000</td>
</tr>
<tr>
<td>50K</td>
<td>50,000</td>
</tr>
<tr>
<td>100K</td>
<td>100,000</td>
</tr>
<tr>
<td>200K</td>
<td>200,000</td>
</tr>
<tr>
<td>300K</td>
<td>300,000</td>
</tr>
<tr>
<td>1M</td>
<td>1,000,000</td>
</tr>
<tr>
<td>2M</td>
<td>2,000,000</td>
</tr>
</tbody>
</table>
Benchmark Dataset Characteristics

• More statistics in file *Dataset Characteristics.xlsx*.

• List of entity profiles in the form of a List<\texttt{EntityProfile}> Java serialized object
  
  – where every object of class \texttt{EntityProfile} corresponds to an individual entity profile.
Structure of the ER-Framework Project

- Block Building Methods
  - Disk-based Methods
  - Memory-based Methods
- Block Processing Methods
  - Block-refinement
  - Comparison-refinement
- Meta-blocking
- Utilities, Data Structures,...
Block Building Methods

• Common interface for all methods imposed by AbstractBlockingMethod.
  – Input: path to the entity collection(s) 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. Need to specify the relative path.
Block Processing/Meta-blocking Methods

- 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 $\text{Set<IdDuplicates>}$ Java serialized object
  – where every object of class `IdDuplicates` contains a pair of entity ids.

• Class `BlockStatistics` measures the performance of a block collection wrt:
  – $\text{PC, PQ, } | |B| |,|D_B|, \text{ BC, CC.}$
Thank You!
Questions?
References – Part A


References – Part B


References – Part C


Additional References

