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 (URLs) are used for the same real-world “entity”? 

However ...

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

<table>
<thead>
<tr>
<th>London</th>
<th>런던</th>
<th>런던</th>
<th>लंडन</th>
<th>لندن</th>
<th>лондон</th>
<th>लंडन</th>
<th>ℓανδαν</th>
<th>Landan</th>
<th>Londona</th>
<th>Londonas</th>
<th>Londoni</th>
<th>Londono</th>
<th>Londra</th>
<th>Londres</th>
<th>Londyn</th>
<th>Lontoo</th>
<th>Loundres</th>
<th>Luân</th>
<th>Đôn</th>
<th>Lundén</th>
<th>Lundínir</th>
<th>Lunnainn</th>
<th>Lunnon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Londain</td>
<td>Londe</td>
<td>Londen</td>
<td>Londen</td>
<td>Londen</td>
<td>Londen</td>
<td>Londinium</td>
<td>London</td>
<td>Londona</td>
<td>Londonas</td>
<td>Londoni</td>
<td>Londono</td>
<td>Londra</td>
<td>Londres</td>
<td>Londyn</td>
<td>Lontoo</td>
<td>Loundres</td>
<td>Luân</td>
<td>Đôn</td>
<td>Lundén</td>
<td>Lundínir</td>
<td>Lunnainn</td>
<td>Lunnon</td>
<td></td>
</tr>
</tbody>
</table>

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, ScaDS, Leipzig, July 2016
However ...

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

London 런던 伦敦 लंडन ロンドン
Londain Londe Londen Londen Londinimum
London Londona Londonas Londoni Londono Londra
Londres Londrez Londyn Lontoo Loundres Luân Đôn
Lunden Lundúnir Lunnainn Lunnon
لندن لندن لندن لوندن لوندین لندن لندن لندن
LondresLondona Londonas Londoni Londono Londra
Londres Londrez Londyn Lontoo Loundres Luân Đôn
Lunden Lundúnir Lunnainn Lunnon
Londres Londona Londonas Londoni Londono Londra
Londres Londrez Londyn Lontoo Loundres Luân Đôn
Lunden Lundúnir Lunnainn Lunnon

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, ScaDS, Leipzig, July 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
- ...

... 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 the London
- Social networks in London
- Pictures and tags about London
- Videos and tags for London
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 Papadakis & Palpanas, ScaDS, Leipzig, July 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**
- **Dirty-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 $E$

$|E|$ entities

Brute-force approach

Blocking

Duplicate Pairs

$|E|$ entities
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**

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
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**

Papadakis & Palpanas, ScaDS, Leipzig, July 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. 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 \textit{one or more} blocking keys.
2. Place into blocks all entities having the \textit{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)

\textbf{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
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, \(s \equiv t\) if:
     - \(\text{JaccardSimilarity}(s.\text{title}, t.\text{title}) > 0.80\) AND
     - \(\text{EditDistance}(s.\text{venue}, t.\text{venue}) < 3\)
   - 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
Performance-wise Categorization

1. **Exact Blocking Methods**
   - Maximize PQ for PC = 100%
   - **Closed**-world assumption
   - E.g., for bibliographical records, \( s \equiv t \) if:
     \[
     \text{JaccardSimilarity}(s.\text{title}, t.\text{title}) > 0.80 \ \text{AND} \ \text{EditDistance}(s.\text{venue}, t.\text{venue}) < 3
     \]
   - 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

   *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]

E → Block Building → Block Cleaning → Comparison Cleaning → B

- Lazy blocking methods
- Block-refinement methods
- Comparison-refinement methods

Proactive blocking methods

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
# 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>Redundancy-negative</strong></td>
<td><strong>Redundancy-neutral</strong></td>
</tr>
<tr>
<td><strong>Schema-based</strong></td>
<td>(Extended) Canopy Clustering</td>
</tr>
<tr>
<td>Standard Blocking</td>
<td>2. MFIBlocks</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>
</tbody>
</table>
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 $\rightarrow$ 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
  - Q-grams Blocking
    - Extended Q-grams Blocking
    - MFIBlocks
      - Extended Q-grams Blocking
      - Canopy Clustering
        - Extended Canopy Clustering
  - Suffix Arrays
    - Extended Suffix Arrays
Overview of Schema-based Methods

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

LAZY BLOCKING METHODS

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
Overview of Schema-based Methods

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

LAZY BLOCKING METHODS

PROACTIVE BLOCKING METHODS

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
Overview of Schema-based Methods

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

LAZY BLOCKING METHODS

PROACTIVE BLOCKING METHODS

Papadakis & Palpanas, ScaDS, Leipzig, July 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:

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

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

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

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

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
Overview of Schema-based Methods

Standard Blocking

- Sorted Neighborhood
  - Extended Sorted Neighborhood
  - MFIBlocks
    - Extended Q-grams Blocking
- Suffix Arrays
  - Extended Suffix Arrays
  - Canopy Clustering
    - Extended Canopy Clustering
Overview of Schema-based Methods
blocks contain entities with similar blocking keys

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

Papadakis & Palpanas, ScaDS, Leipzig, July 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 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

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

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
Overview of Schema-based Methods
blocks contain entities with **same, or similar** blocking keys

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

MFIBlocks

Papadakis & Palpanas, ScaDS, Leipzig, July 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

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

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

- Standard Blocking
  - Sorted Neighborhood
    - Extended Sorted Neighborhood
      - Extended Q-grams Blocking
  - Suffix Arrays
    - Extended Suffix Arrays

- Q-grams Blocking
  - MFIBlocks
    - Extended Canopy Clustering
      - Extended Canopy Clustering

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
Overview of Schema-based Methods
blocks contain entities with similar blocking keys
Canopy Clustering [McCallum et. al., KDD 2000]

Blocks on similarity of BKs.

- 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 $T1$ from the center point are a part of the canopy but can also be canopy centers themselves.
- Center of the Canopy($P$).
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
  - Suffix Arrays
    - Extended Suffix Arrays
- Q-grams Blocking
  - MFIBlocks
    - Extended Q-grams Blocking
  - Canopy Clustering
    - Extended Canopy Clustering

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
Overview of Schema-based Methods
blocks contain entities with **same** blocking keys

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

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
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 \( l_{\text{min}} \).
For \( l_{\text{min}} = 3 \), 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 $l_{\text{min}}$ 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 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

DATASET 1

Entity 1
- name=United Nations Children’s Fund
- acronym=unicef
- headquarters=California
- address=Los Angeles, 91335

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, ScaDS, Leipzig, July 2016
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 $\rightarrow$ 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
- hdq=California
- status=active

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

Papadakis & Palpanas, ScaDS, Leipzig, July 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 → smaller blocks, higher efficiency.

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
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 $\text{sim}(n_i, n_j) > 0$, 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
   – 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 $\vartheta$).

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.

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
For Semantic Web data, three sources of evidence create blocks of lower redundancy than Token Blocking:

1. Infix

2. Infix Profile

3. Literal Profile

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 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**
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

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
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

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
Distributions of Block Sizes and Duplicates

1.0E+00  1.0E+03  1.0E+06  1.0E+09  1.0E+12

1E+0   1E+1   1E+2   1E+3   1E+4   1E+5   1E+6

1E+0   1E+4   1E+8   1E+12

Number of Blocks

% of Duplicates

Block Cardinality

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
Distributions of Block Sizes and Duplicates

Papadakis & Palpanas, ScaDS, Leipzig, July 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
Main idea:
• restrict the size of every block into $[b_{\text{min}}, b_{\text{max}}]$ 
  – 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 $b_{\text{min}}$ 
  – 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. 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 $\Rightarrow 3.7$ 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 $\Rightarrow 3.7$ 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
- Edge Weighting
- Graph Pruning
- Block Collecting

Papadakis & Palpanas, ScaDS, Leipzig, July 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!
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
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, \( k = BPE - 1 \)

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
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)
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 → \textbf{3.7 years}

Token Blocking + Block Filtering + Comparison Propagation
Overhead time: <30 mins
Comparisons: 3.5 \cdot 10^{10}
Recall: 99%
Total Running time: \textbf{19 days}

Token Blocking + Block Filtering + \textbf{Meta-blocking}
Overhead time: 4 hours
Comparisons: 8.95 \cdot 10^{6}
Recall: 99%
Total Running time: \textbf{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

```
Key   Value
  e1  b1,b4,b6
  e2  b1
  e3  b1,b4
  e4  b4,b5
  ... ...
```

```
Map
```

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

```
Group by key
```

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

```
Reduce
```

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

```
Reduce
```

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

```
Reduce
```

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

```
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
```

```
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
```

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
Meta-blocking (advanced)
Weighted Edge Pruning (WEP) & Jaccard Scheme (JS)

Key Value

Map → Key Value

Key Value

Map → Group by key

Reduce

Map → Reduce

Reduce

Reduce

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
### Meta-blocking (advanced)

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

<table>
<thead>
<tr>
<th></th>
<th>( DB_C )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Basic</td>
</tr>
<tr>
<td>Block Filtering</td>
<td>2</td>
</tr>
<tr>
<td>CEP</td>
<td>CBS</td>
</tr>
<tr>
<td></td>
<td>ECBS</td>
</tr>
<tr>
<td></td>
<td>JS</td>
</tr>
<tr>
<td>CNP</td>
<td>CBS</td>
</tr>
<tr>
<td></td>
<td>ECBS</td>
</tr>
<tr>
<td></td>
<td>JS</td>
</tr>
<tr>
<td>WEP</td>
<td>CBS</td>
</tr>
<tr>
<td></td>
<td>ECBS</td>
</tr>
<tr>
<td></td>
<td>JS</td>
</tr>
<tr>
<td>WNP</td>
<td>CBS</td>
</tr>
<tr>
<td></td>
<td>ECBS</td>
</tr>
<tr>
<td></td>
<td>JS</td>
</tr>
</tbody>
</table>
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
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 $\rightarrow$ several blocking keys (tokens) $\rightarrow$ multiple attribute names $\rightarrow$ aggregate entropy $\cdot$ Pearson’s $\chi^2$

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):
      $$\theta_i = \arg\max_i w(e_{ij})$$
   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

![Diagram]

E → Block Building (Lazy blocking methods) → Block Cleaning (Block-refinement methods) → Comparison Cleaning (Comparison-refinement methods)

- Proactive blocking methods
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

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
Experimental Analysis Setup

• Exhaustive parameter tuning to identify two configurations for each method:
  1. Best configuration per dataset $\rightarrow$ maximizes
     \[ a(B, E) = RR(B, E) \cdot PC(B, E) \]
  2. Default configuration $\rightarrow$ highest average $a$ 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

Token-blocking and Meta-blocking
Time Efficiency of Lazy Methods on DBPedia

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
Time Efficiency of Lazy Methods on DBPedia

Token-blocking and Meta-blocking

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
Effectiveness of Proactive methods on DBPedia
Effectiveness of Proactive methods on DBPedia

- Suffix-arrays and Meta-blocking

Graph showing the comparison of different methods in terms of \|B\| and PC.
Time Efficiency of Proactive Methods on DBPedia

- Graph 1: OTime
- Graph 2: RTime

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
Time Efficiency of Proactive Methods on DBPedia

Suffix-arrays and Meta-blocking

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
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
get most of the benefit much earlier
Progressive Blocking

Facts:

- Progressive, or Pay-as-you-go ER comes is useful to get most of the benefit much earlier.

- May require some pre-processing.

Papadakis & Palpanas, ScaDS, Leipzig, July 2016
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 \(\rightarrow\) 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:

**Step 1: Data Reading**
- Reads files containing the entity profiles and the golden standard.

**Step 2: Block Building**
- Creates overlapping blocks.

**Step 3: Block Cleaning**
- Optional step that cleans blocks from useless comparisons (repeated, superfluous).

**Step 4: Comparison Cleaning**
- Optional step that operates on the level of individual comparisons to remove the useless ones.

**Step 5: Entity Matching**
- Executes all retained comparisons.

**Step 6: Entity Clustering**
- Partitions the similarity graph into equivalence clusters.

**Step 7: Evaluation & Storing**
- Stores and presents performance results w.r.t. numerous measures.
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:

Step 1: Data Reading
Possible to read CSV, RDF/XML files & relational DBs in any combination!

Step 2: Block Building
Choose 1 out of 8 methods.

Step 3: Block Cleaning
Specify any combination of 3 (4) complementary methods for Dirty (Clean-Clean) ER.

Step 4: Comparison Cleaning
Choose 1 out of 7 methods (including Meta-blocking).

Step 5: Entity Matching
Combine 1 out of 2 methods with 12 textual representation models and 10 similarity measures.

Step 6: Entity Clustering
Choose 1 out of 6 methods for Dirty ER. For Clean-Clean ER, 1 method is available.

Step 7: Evaluation & Storing
Store results as a CSV file.
### Which Blocking Methods are included?

<table>
<thead>
<tr>
<th>Block Building</th>
<th>Block Cleaning</th>
<th>Comparison Cleaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token Blocking</td>
<td>Block Filtering</td>
<td>Comparison Propagation</td>
</tr>
<tr>
<td>Sorted Neighborhood</td>
<td>Size-based Block Purging</td>
<td>Cardinality Edge Pruning (CEP)</td>
</tr>
<tr>
<td>Extended Sorted Neighborhood</td>
<td>Cardinality-based Block Purging</td>
<td>Cardinality Node Pruning (CNP)</td>
</tr>
<tr>
<td>Attribute Clustering</td>
<td>Block Scheduling</td>
<td>Weighted Edge Pruning (WEP)</td>
</tr>
<tr>
<td>Q-Grams Blocking</td>
<td></td>
<td>Weighted Node Pruning (WNP)</td>
</tr>
<tr>
<td>Extended Q-Grams Blocking</td>
<td></td>
<td>Reciprocal CNP</td>
</tr>
<tr>
<td>Suffix Arrays</td>
<td></td>
<td>Reciprocal WNP</td>
</tr>
<tr>
<td>Extended Suffix Arrays</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Where can I find JedAI Toolkit?


  - All code is implemented using Java 8.
  - All code is publicly available under Apache License V2.0.


- When using JedAI, please cite:

Which datasets are available for testing?


<table>
<thead>
<tr>
<th>Clean-Clean ER (real)</th>
<th>D1 Entities</th>
<th>D2 Entities</th>
<th>Dirty ER (synthetic)</th>
<th>Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abt-Buy</td>
<td>1,076</td>
<td>1,076</td>
<td>10K</td>
<td>10,000</td>
</tr>
<tr>
<td>DBLP-ACM</td>
<td>2,616</td>
<td>2,294</td>
<td>50K</td>
<td>50,000</td>
</tr>
<tr>
<td>DBLP-Scholar</td>
<td>2,516</td>
<td>61,353</td>
<td>100K</td>
<td>100,000</td>
</tr>
<tr>
<td>Amazon-GP</td>
<td>1,354</td>
<td>3,039</td>
<td>200K</td>
<td>200,000</td>
</tr>
<tr>
<td>Movies</td>
<td>27,615</td>
<td>23,182</td>
<td>300K</td>
<td>300,000</td>
</tr>
<tr>
<td>DBPedia</td>
<td>1,190,733</td>
<td>2,164,040</td>
<td>1M</td>
<td>1,000,000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2M</td>
<td>2,000,000</td>
</tr>
</tbody>
</table>

Can be used for Dirty ER, as well.
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


References – Part B


References – Part C


