Sorted Neighborhood for Schema-free RDF Data

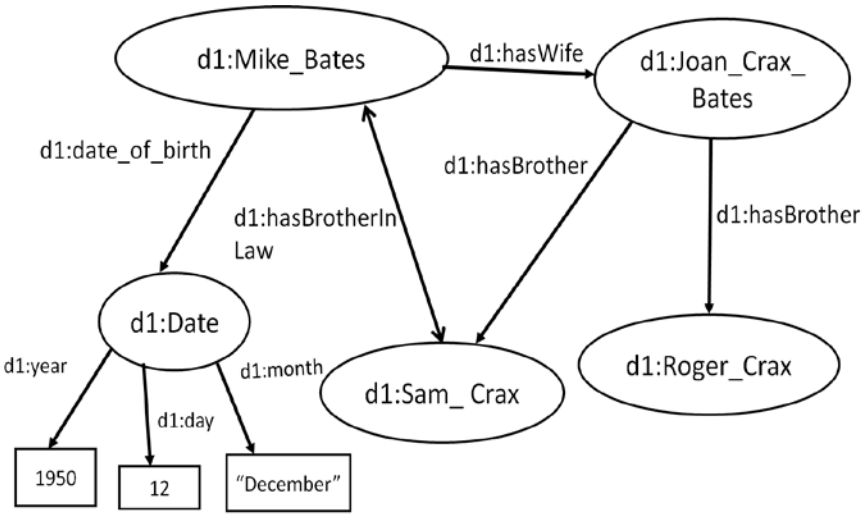
Mayank Kejriwal, Daniel P. Miranker

University of Texas at Austin

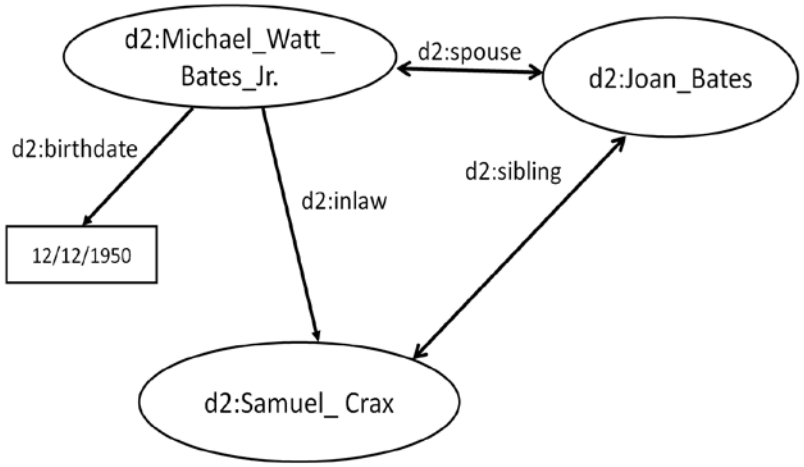
Acknowledgement: National Science Foundation

Instance Matching

- **High-level question**: Which instances refer to the same underlying entity?



RDF Graph 1

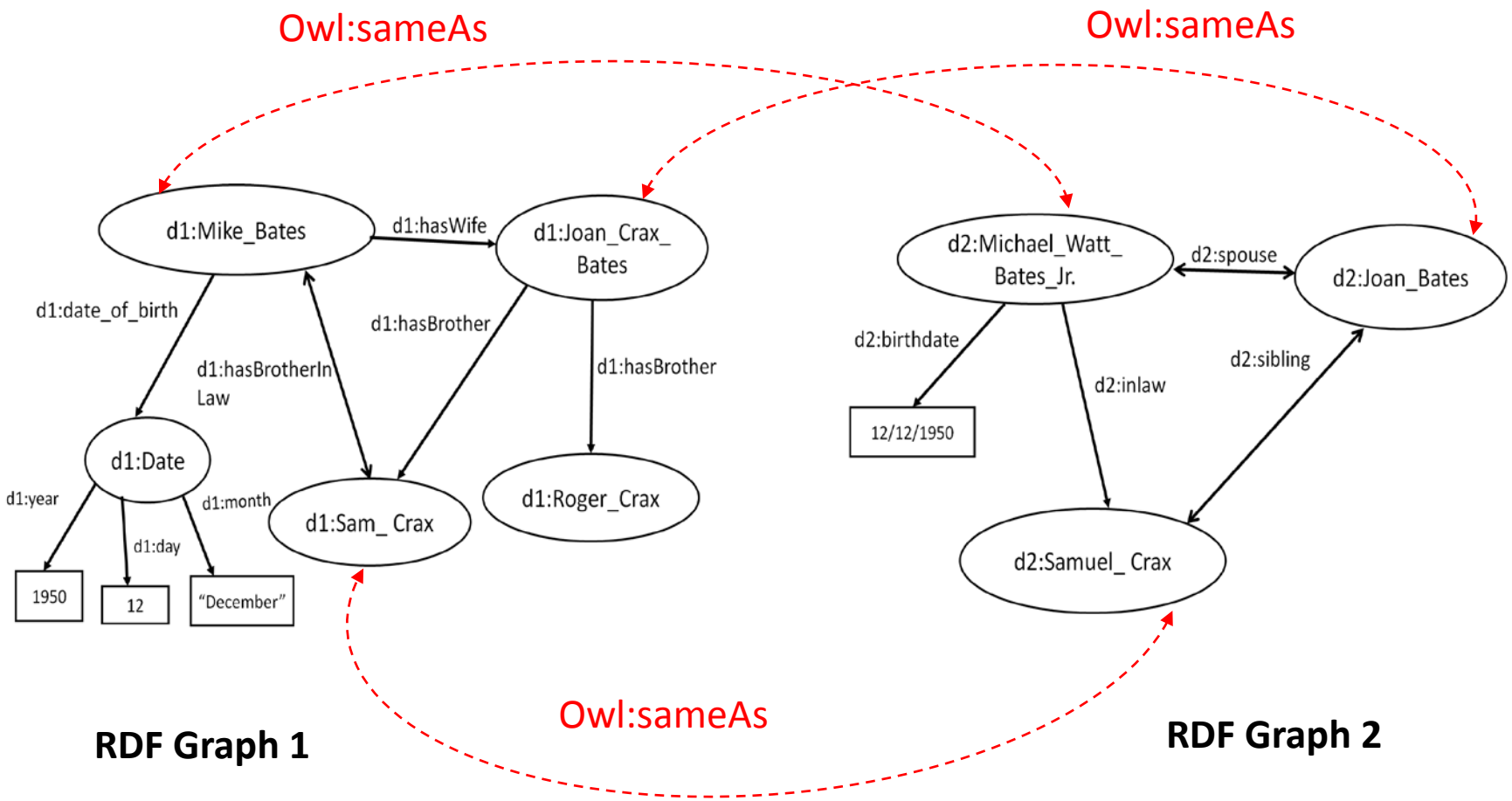


RDF Graph 2

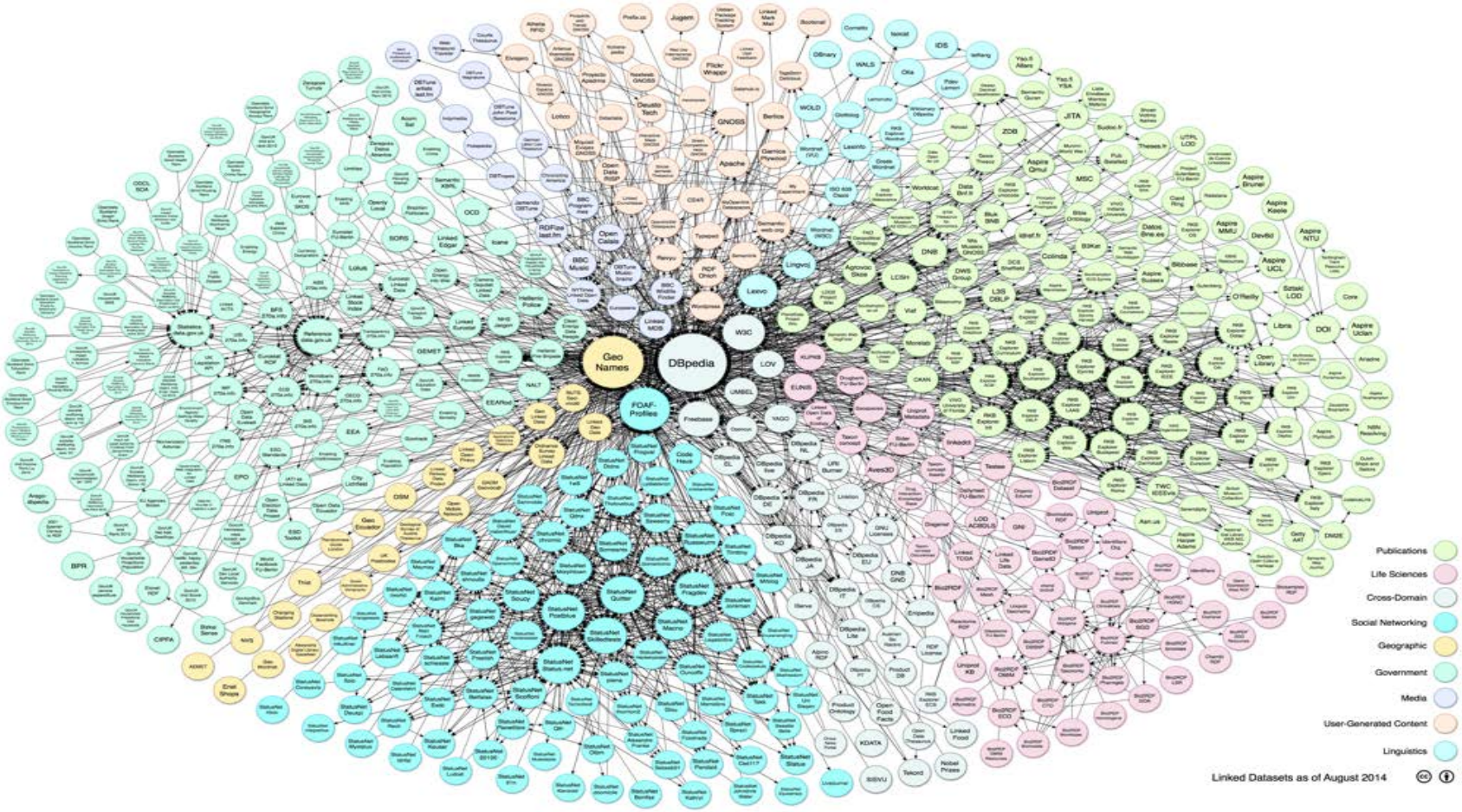
- Usually easy for a **human being**...

Jaffri et al. (2008)
Nikolov et al. (2011)

- Instance Matching: a 50 year old **Artificial Intelligence** problem!



Primary Application: Linked Open Data

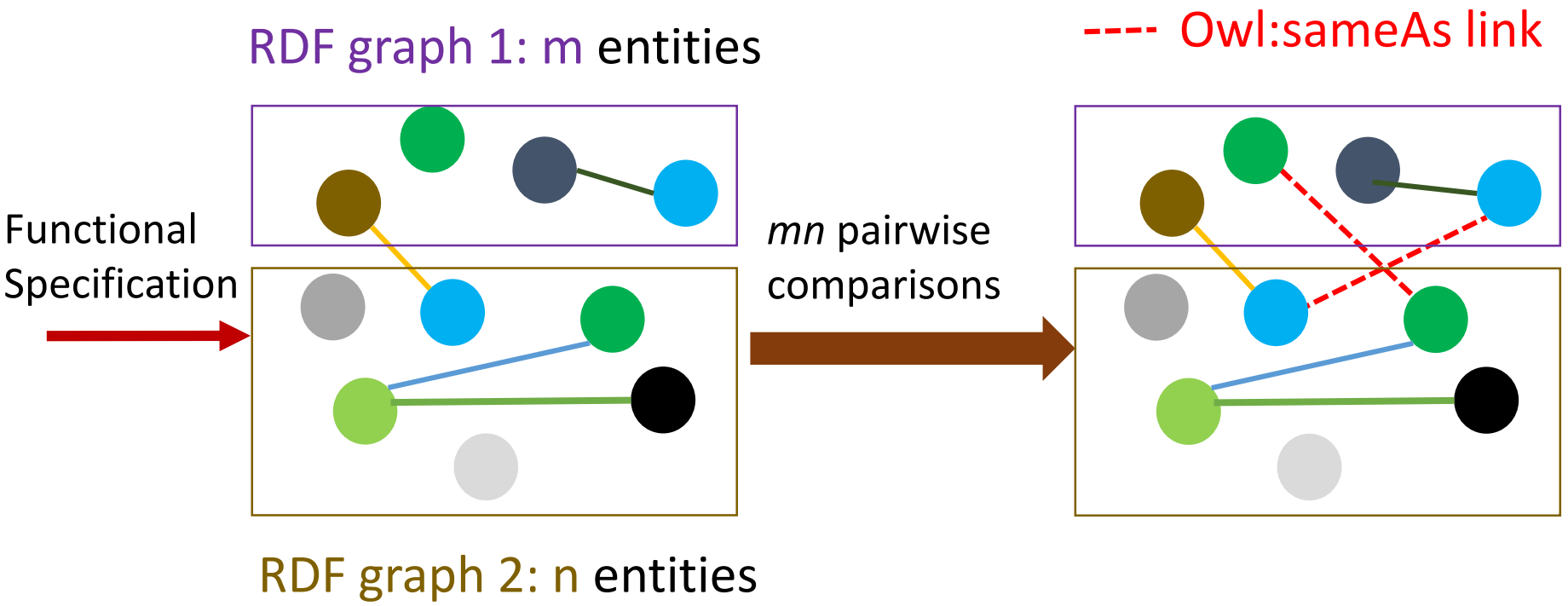


- **Key Challenge:** Efficient instance matching on many schema-free RDF datasets

Linkeddata.org
Bizer et al.

Efficient Instance Matching

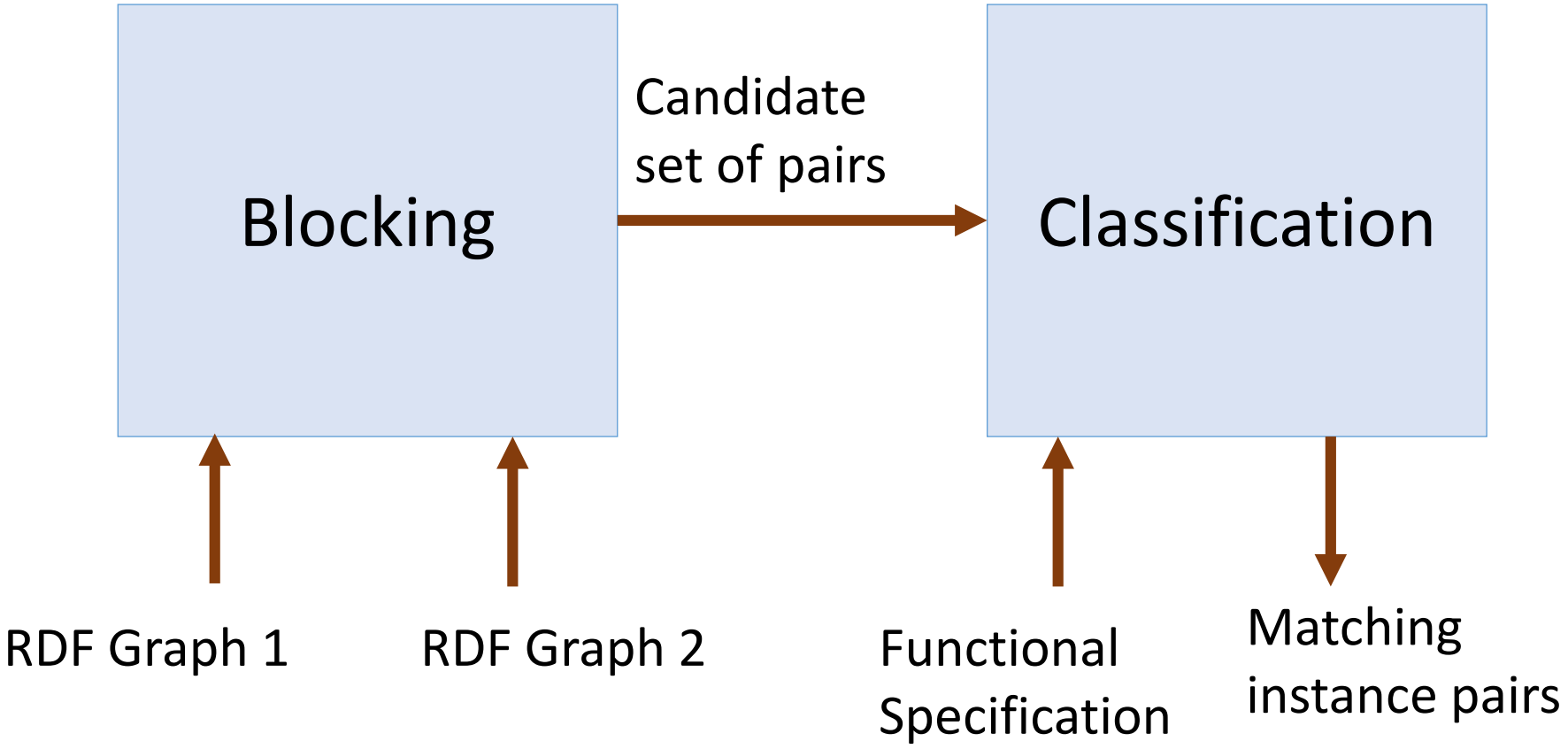
How do we locate **matching** instance pairs without comparing **all** pairs?



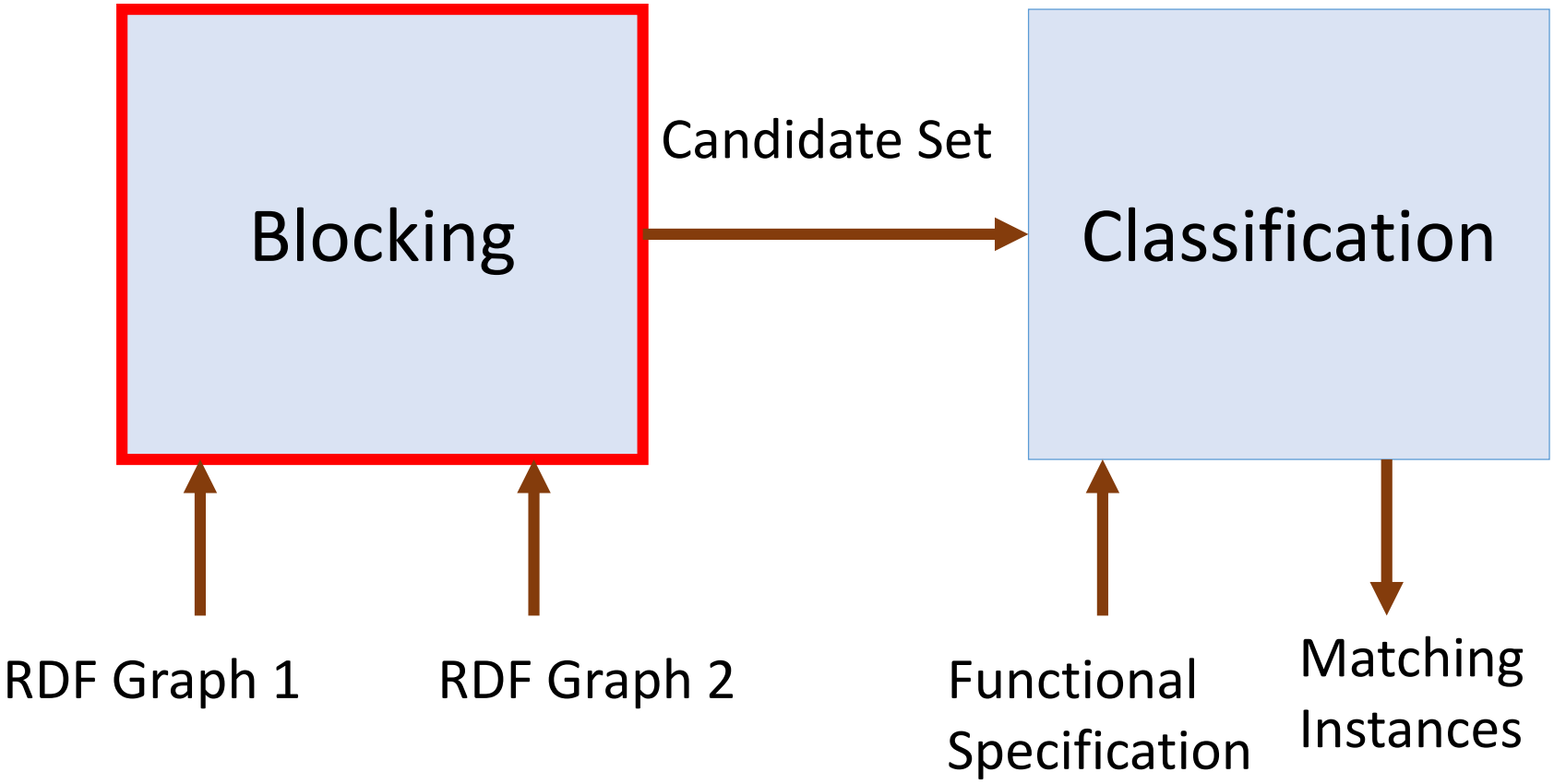
Instead, choose a small subset of the *nm* pairs that are good link candidates in a blocking (similar to clustering) phase

Christen (2012)

A Two-step Instance Matcher



Elmagarmid et al. (2007)

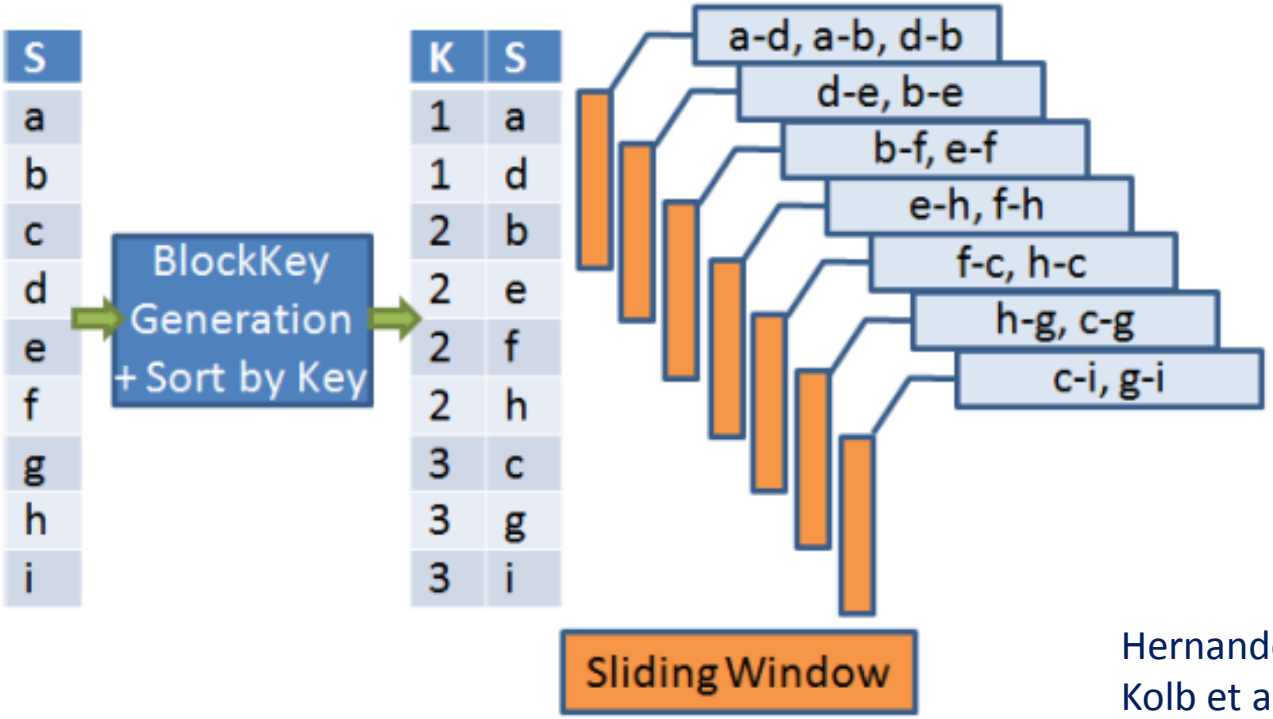


This work adapts a classic blocking procedure called **Sorted Neighborhood** for **schema-free RDF** data

Hernandez and Stolfo (1995)

Traditional Sorted Neighborhood

- Proposed for tabular databases in 1995
- Assumes a heuristic called a *blocking key* (e.g. *Tokens(Last-Name*)
- Many variants proposed (including for XML databases)



Hernandez and Stolfo (1995)
Kolb et al. (2011)

How do we implement SN for schema-free RDF use-cases?

- Non-trivial!
  - By definition, the data has no schema, which Sorted Neighborhood presumes
  - Data may be accessible either through a batch dump or through a SPARQL endpoint

- This work proposes a Sorted Neighborhood workflow to address these challenges

Key Intuitions

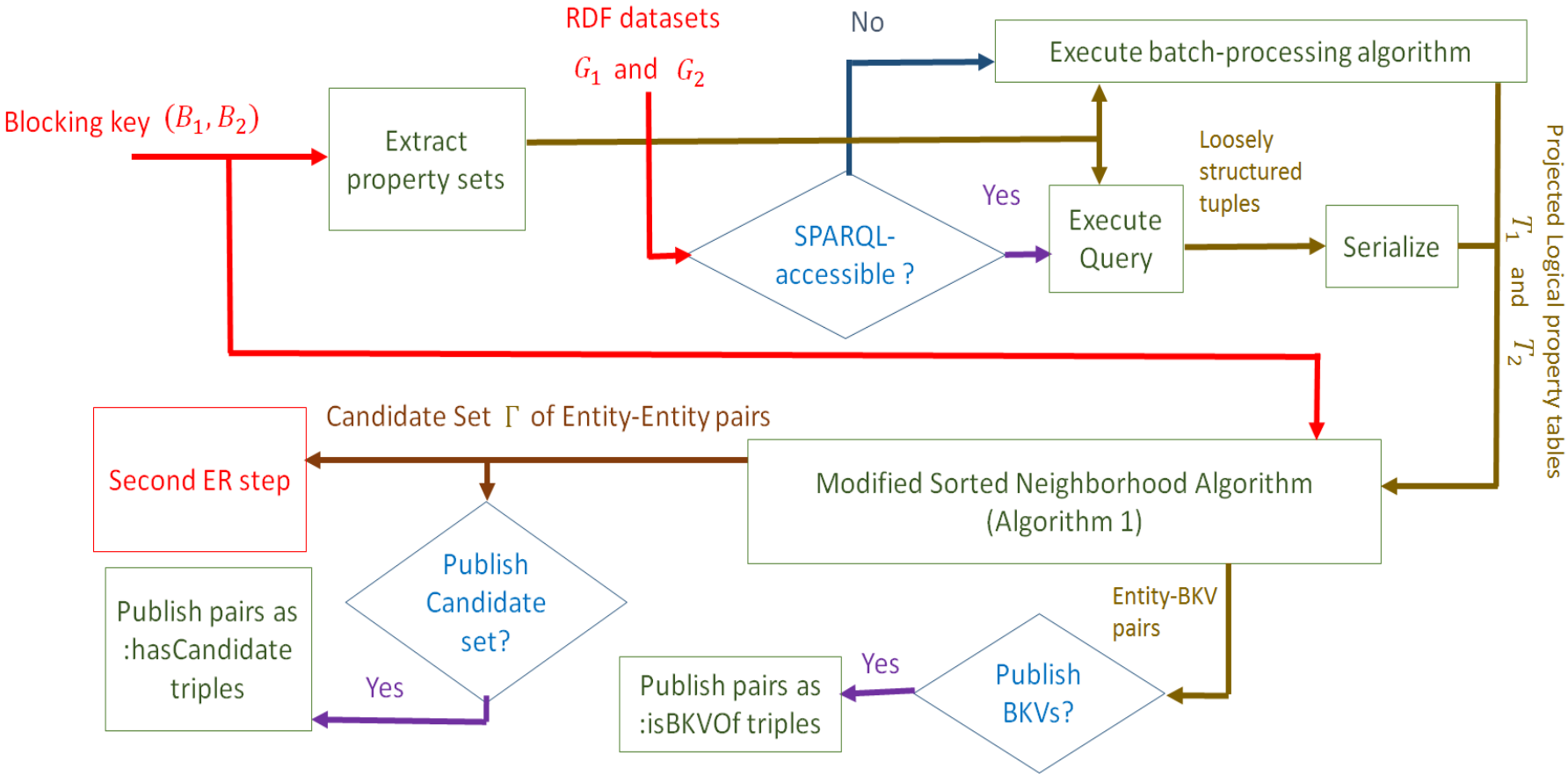
- We will construct a 'pseudo-schema' from the properties in the dataset
  - Use a logical property table for representing the RDF graph

Subject	Property 1	Property 2	...	Property n
Subject URI 1				
Subject URI 2				
...				

- Run a modified Sorted Neighborhood algorithm on the two property tables
  - Original algorithm only assumes one table and schema

Wilkinson et al. (2003)

Proposed Workflow



Constructing the SPARQL query

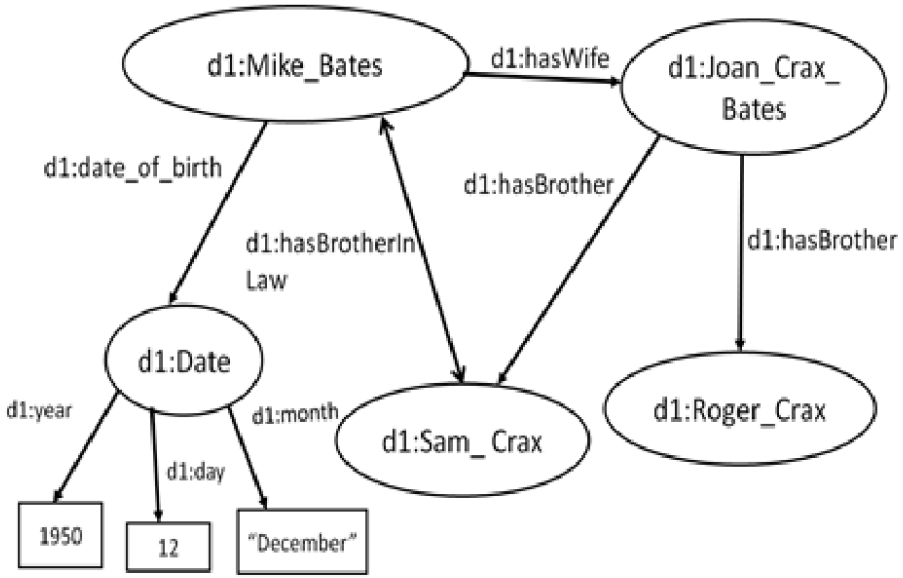
- Assume that the blocking key for each RDF dataset is defined only in terms of the properties in that dataset
  - Let the property set constitute the set of properties in the blocking key

SELECT ?entity $?o_1 \ldots ?o_d$
WHERE{
{SELECT DISTINCT ?entity WHERE ?entity ?p ?o.
FILTER (?p = $< p_1 >$ || ... || ?p = $< p_d >$}}
OPTIONAL {?entity $< p_1 >$ $?o_1$}
...
OPTIONAL {?entity $< p_n >$ $?o_d$}
}

Example: Constructing the SPARQL query

Assume the blocking key Tokens(subject) U Tokens(d1:hasBrother)

Run the **instantiated** query



Subject	**d1:hasBrother**
d1:Mike_Bates	
d1:Joan_Crax_Bates	d1:Sam_Crax
d1:Joan_Crax_Bates	d1:Roger_Crax
d1:Date	
d1:Sam_Crax	

Loosely Structured Tuples

Property Table

Subject	d1:hasBrother
d1:Mike_Bates	
d1:Joan_Crax_Bates	d1:Sam_Crax
d1:Joan_Crax_Bates	d1:Roger_Crax
d1:Date	
d1:Sam_Crax	

Loosely Structured Tuples

Subject	d1:hasBrother
d1:Mike_Bates	*null*
d1:Joan_Crax_Bates	d1:Sam_Crax; d1:Roger_Crax
d1:Date	*null*
d1:Sam_Crax	*null*

Logical Property Table

Wilkinson et al. (2003)

Modified Sorted Neighborhood

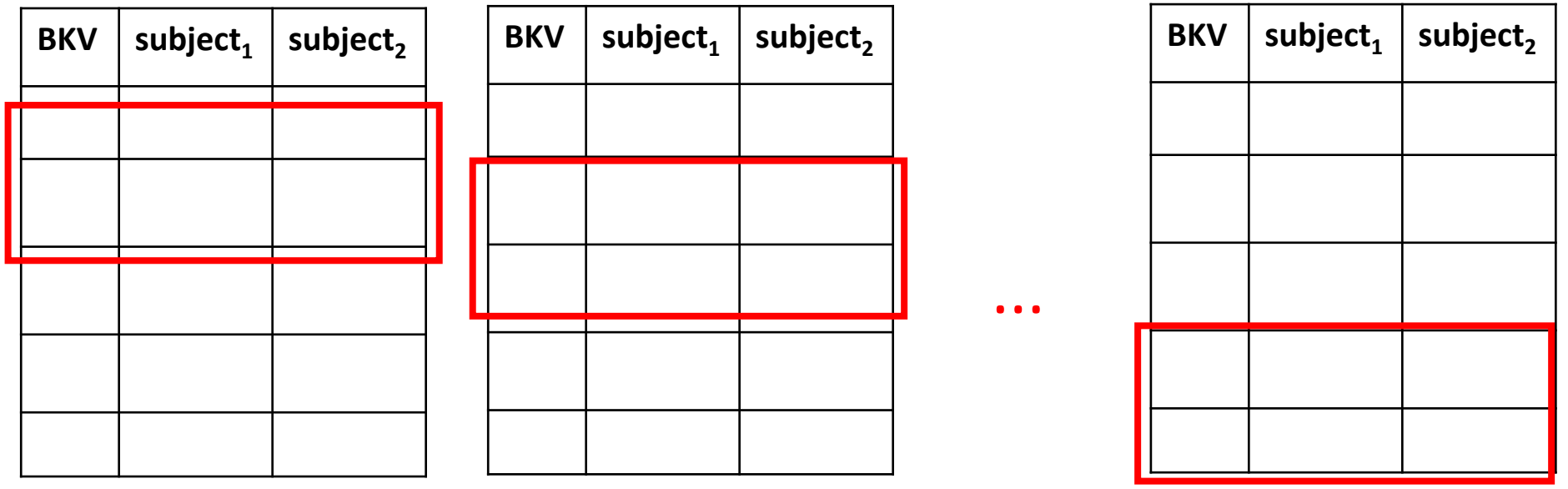
Algorithm 1 Modified Sorted Neighborhood

Input: Blocking keys $B_{1,2}$, projected logical property tables $T_{1,2}$, windowing parameter w

Output: Candidate set Γ of entity-entity pairs

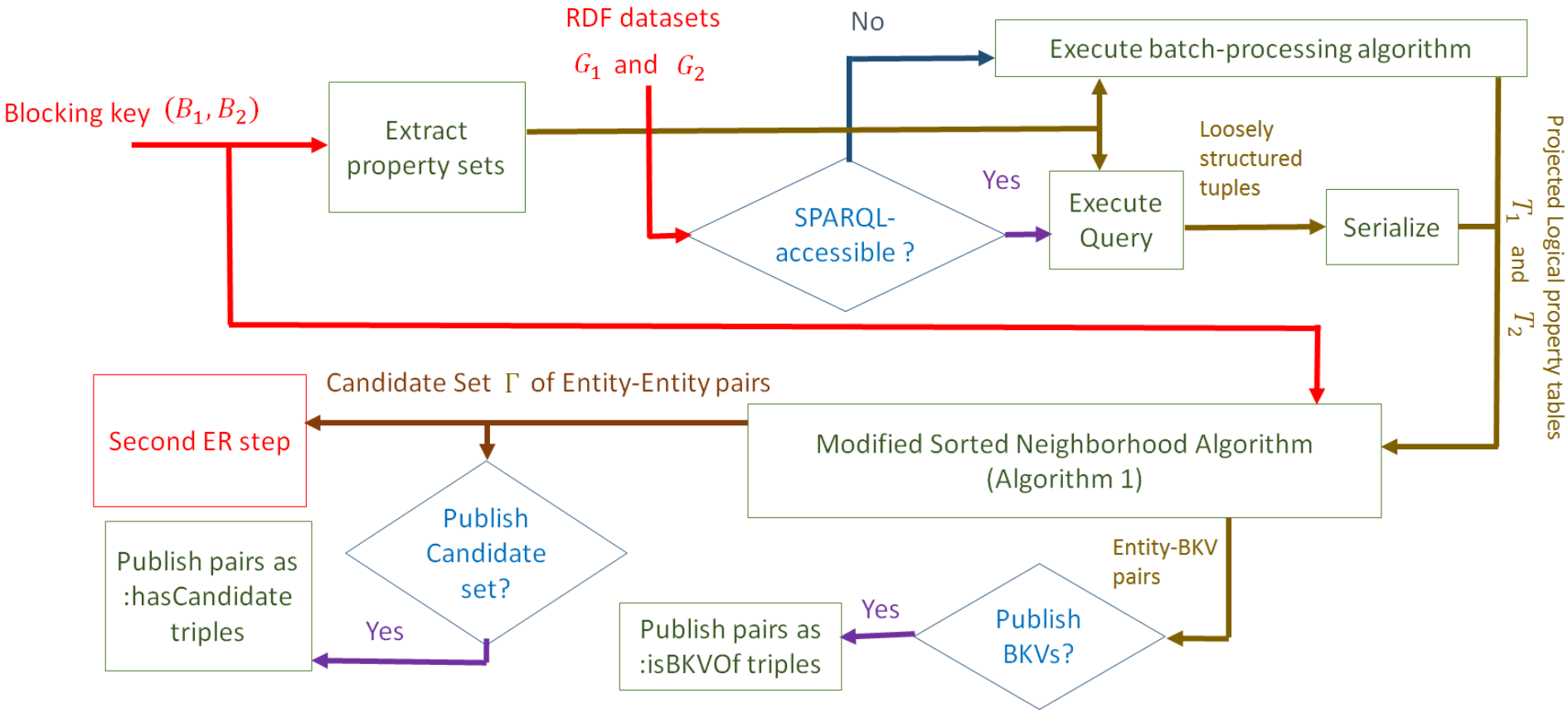
1. Initialize Γ to be an empty set
2. Apply B_1 (B_2) to each tuple in T_1 (T_2) to obtain *multimaps* $\Pi_1 = \{< bkv, R >\}$ ($\Pi_2 = \{< bkv, S >\}$), with bkv in each key-value set pair referring to a blocking key value string and R (S), denoted as a *block*, being a set of subject URIs in T_1 (T_2)
3. Join Π_1 and Π_2 on their keys to obtain joined map $\Pi_{map} = \{< bkv, < R, S >\}$, where $keySet[\Pi_{map}] = keySet[\Pi_1] \cap keySet[\Pi_2]$, and the larger of R and S in each pair in Π_{map} is truncated so that $|R| = |S|$
4. Sort Π_{map} into a list L with columns BKV, $subject_1$ and $subject_2$, using the keys in Π_{map} (or the BKV column) as sorting keys
5. For each tuple $< bkv, s_1, s_2 >$ in L, emit pairs $< s_1, bkv >$ and $< s_2, bkv >$ as entity-BKV pairs (Figure 2)
6. Slide a window of size w over tuples in L (Figure 4)
7. Add entity-entity pair $< e_1, e_2 >$ to Γ, where e_1 (e_2) is a subject URI from column $subject_1$ ($subject_2$) and e_1 and e_2 are in (not necessarily the same) tuples that fall within a common window
8. **return** Γ

Example: Sliding window with w=2



Hernandez and Stolfo (1995)

Recap: proposed system



Experiments: Datasets and Baselines

ID	Name	Triples	Entity pairs	Matching entities
1	Persons 1	9000/7000	$2000 \times 1000 = 2$ million	500
2	Persons 2	10,800/5600	$2400 \times 800 \approx 1.92$ million	400
3	Restaurants	1130/7520	$339 \times 2256 = 764{,}784$	89
4	IM-Similarity	2204/2184	$181 \times 180 = 32{,}580$	496
5	Film	9995/8979	$1549 \times 519 = 803{,}931$	412

- The blocking key was learned by using the *Attribute Clustering* method
- Using the key, we execute Sorted Neighborhood and tune window size *w* for optimal performance.
- We compared against two baselines:
  - *Block purging*
  - *Canopies*

Papdakis et al. (2011)
McCallum et al. (2000)

Experiments: Metrics

- Let *C* be the size of the candidate set output by the blocking method, and *T* be the size of the ground-truth set of true positives. Let O be the size of the Cartesian product of all pairs

$$Pairs\ Completeness\ (PC) = \frac{C\ \cap T}{T}$$

$$Reduction\ Ratio = 1 - \frac{C}{O}$$

$$F - Score = \frac{2 * PC * RR}{PC + RR}$$

Christen (2012)

Results

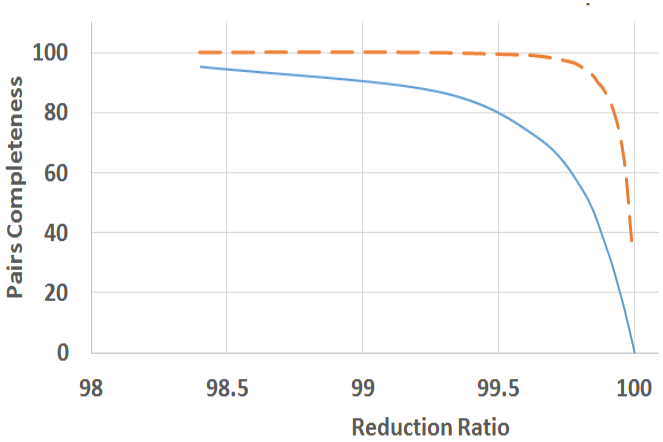
	Test Case	Sorted Neighborhood			Block Purging		
		PC	RR	F-score	PC	RR	F-score
1	Persons 1	100.00%	99.26%	**99.63%**	100.00%	98.86%	99.43%
2	Persons 2	91.25%	89.77%	90.50%	99.75%	99.02%	**99.38%**
3	Restaurants	100.00%	99.41%	99.70%	100.00%	99.57%	**99.79%**
4	IM-Similarity	100.00%	84.73%	**91.73%**	100.00%	62.79%	77.14%
5	Film	97.33%	92.10%	**94.64%**	97.33%	73.09%	83.49%
	Average	*97.72%*	**93.05%**	**95.24%**	**99.42%**	*86.67%*	*91.85%*
	Count	*4/0/1*	*0/3/2*	*0/3/2*	*4/1/0*	*0/2/3*	*0/2/3*

- Sorted Neighborhood provides more stable performance
- Small differences in RR are known to have large impact on run-time, due to quadratic dependence of RR on the number of entities
- Canopies outperforms both methods on average RR, but is outperformed by both methods on PC and F-Score.

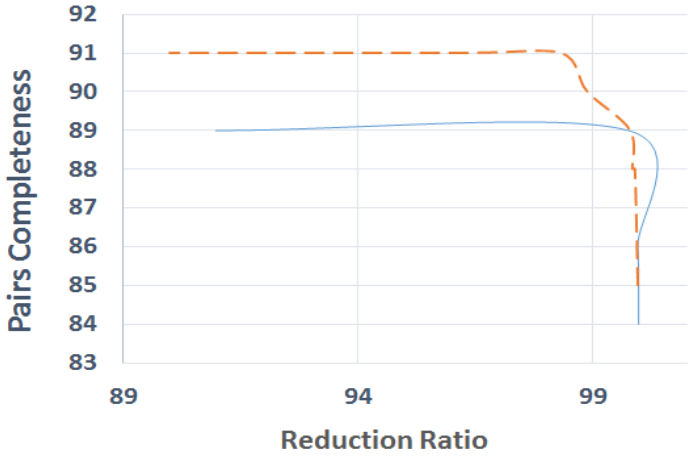
Conclusion

- Efficient Instance Matching on schema-free RDF data is a challenging and important problem
  - Linked Open Data has many schema-free datasets
- Sorted Neighborhood is a classic algorithm with many variants, but has mainly been applied to Relational and XML Databases with schema
- We present a flexible Sorted Neighborhood workflow for schema-free RDF data

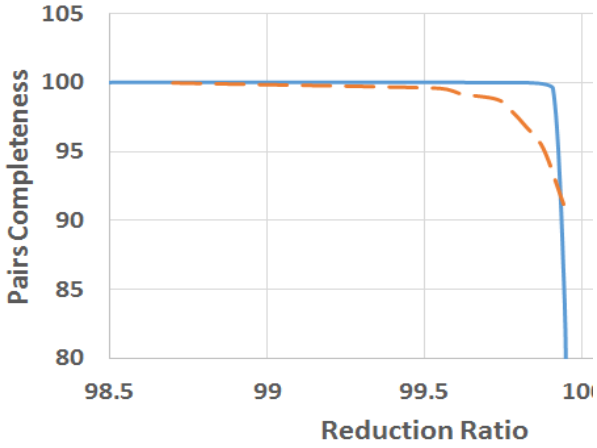
More Results (*Canopies* baseline)



Video Game



Restaurant



Persons