Sorted Neighborhood for the Semantic Web

Mayank Kejriwal, Daniel P. Miranker
University of Texas at Austin
{kejriwal,miranker}@cs.utexas.edu

Abstract

Entity Resolution (ER) concerns identifying logically equivalent entity pairs across databases. To avoid $\Theta(n^2)$ pairwise comparisons of n entities, blocking methods are used. Sorted Neighborhood is an established blocking method for relational databases. It has not been applied on graph-based data models such as the Resource Description Framework (RDF). This poster presents a modular workflow for applying Sorted Neighborhood to RDF. Real-world evaluations demonstrate the workflow's utility against a popular baseline.

Entity Resolution (ER) is the abstract problem of identifying pairs of entities across databases that are syntactically disparate but logically equivalent. The problem goes by multiple names in the AI community, examples being *record linkage*, *instance matching*, and *coreference resolution* (Elmagarmid, Ipeirotis, and Verykios 2007).

The dominant data model in the Semantic Web is the graph-based *Resource Description Framework* (RDF) model. In RDF[1], entities are represented by nodes and edges denote properties between nodes (Figure 1).

In Figure 1, *Cathy Ridley* is a single logical entity but has two syntactic mentions. The goal of ER is to link such pairs of logically equivalent mentions using *sameAs* edges (Ferraram, Nikolov, and Scharffe 2013).

Given n entities and a sophisticated similarity function that determines whether an entity pair should be linked using *sameAs*, a naïve ER application would be quadratic. Scalability indicates a two-step approach. The first step is a *blocking method*, which clusters entities into possibly overlapping *blocks*, such that entities sharing a block are likely to be duplicates, and merit further similarity comparisons. Entities not sharing a block are never compared, leading to savings (Christen 2012).

In the relational database community, blocking methods are well-researched (Christen 2012), with a popular approach being *Sorted Neighborhood* (SN) (Hernández and Stolfo 1995). Despite excellent empirical performance, SN has never been applied to *linked data* (Schmachtenberg, Bizer, and Paulheim 2014). Linked Open Data (LOD) continues to grow and currently contains billions of resources[2]. Linked data is often accessed through a *SPARQL-endpoint*, where SPARQL[3] is a declarative pattern-matching query language.

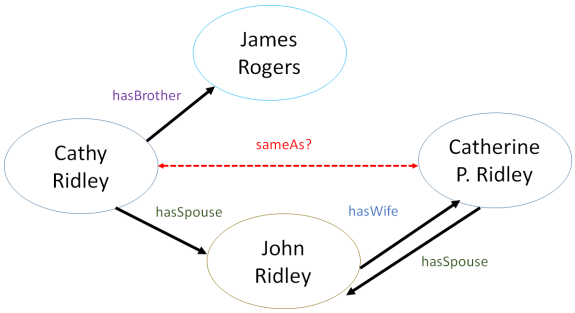

Figure 1: A simple instance of ER in an RDF graph

Table 1: Tuples sorted using blocking key values (BKVs)

ID	First Name	Last Name	Zip	BKV
1	Cathy	Ransom	77111	CR7
2	Catherine	Ridley	77093	CR7
3	Cathy	Ridley	77093	CR7
4	John	Rogers	78751	JR7
5	J.	Rogers	78732	JR7
6	John	Ridley	77093	JR7
7	John	Ridley Sr.	77093	JRS7

Blocking has recently become an active area of research in the Semantic Web community (Isele, Jentzsch, and Bizer 2011), but SN has been challenging to apply to linked data. One reason is that RDF data can be *schema-free* and may not include *type* information (especially on Linked Open Data), while SN explicitly assumes *structured tuples*. Consider Table 1 for an application of SN. First, a provided *blocking key* is used to extract blocking key *values* (BKVs) from tuples. Using + as the concatenation operator, the blocking key applied on Table 1 was *Initials(First Name)+Initials(Last Name)+Initials(Zip)*. Next, BKVs are used as sorting keys.

[1] www.w3.org/RDF

[2] linkeddata.org

[3] www.w3.org/TR/rdf-sparql-query/

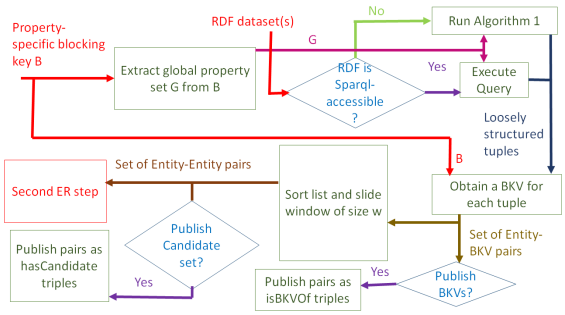

Figure 2: A linked data Sorted Neighborhood workflow

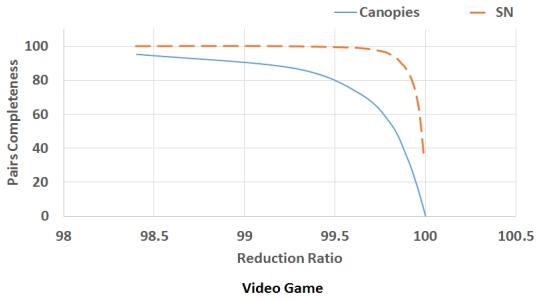

Figure 3: Results on the *Video Game* benchmark

Finally, a window of (pre-determined) constant size w is slid over the sorted tuples. Entities sharing a window are paired with each other and become candidates for further evaluation. Typical values of w range from 2 to 10. Assuming $w << n$, the sorting cost is found to dominate SN run-time (Hernández and Stolfo 1995).

SN is considered state-of-the-art among relational practitioners, owing to its excellent performance. New SN variants continue to be researched and evaluated, but are limited to relational databases (Draisbach and Naumann 2009). One possible reason why linked data has proved to be a challenging application for SN is because it is *distributed*, with many datasets only accessible using SPARQL. As such, it is not evident how to *operationalize* SN for such data.

An SN workflow that can operate on linked data available either as downloadable dumps or through a SPARQL endpoint is presented in Figure 2. The full technical details are in an expanded work[4]. First, we propose a specific class of blocking keys called *property-specific blocking keys*. This class of blocking keys is schema-agnostic and may be applied on semi-structured graph data rather than structured tuples. Intuitively, these keys treat the properties of an RDF graph like the columns of a table. An example (on Figure 1) would be *Tokens(hasSpouse)+Initials(Name)*. The keys can be framed in terms of a SPARQL query if the data is not available as a dump. If it is, a linear-time algorithm is executed. The output of either option is a set of *loosely-structured* tuples on which the property-specific blocking key B can operate to produce BKVs. By virtue of a sliding-window process, a candidate set of promising entity-entity pairs can be populated and input to the second ER step. The workflow offers options to data publishers and administrators to publish intermediate outputs in the form of *third-party* triples. Such triples can be used as a cache to appropriately distribute the load for large datasets, for example.

In Figure 3, experimental results from the *Video Game* benchmark are provided. This benchmark contains over 230,000 triples, and is real-world. The proposed method is shown to outperform an established clustering baseline, *Canopies* (McCallum, Nigam, and Ungar 2000). Reduction Ratio and Pairs Completeness are standard metrics that respectively measure *efficiency* and *effectiveness* of blocking (Christen 2012).

In the expanded work, we also show results from two other real-world test cases. Qualitatively, these results demonstrate that the workflow can adapt gracefully to RDF and that SN benefits need not be restricted merely to the tabular domain. In future work, we will investigate extensions of the workflow, as well as a MapReduce implementation.

[4]https://sites.google.com/a/utexas.edu/mayank-kejriwal/projects/sorted-neighborhood

References

Christen, P. 2012. A survey of indexing techniques for scalable record linkage and deduplication. *Knowledge and Data Engineering, IEEE Transactions on* 24(9):1537–1555.

Draisbach, U., and Naumann, F. 2009. A comparison and generalization of blocking and windowing algorithms for duplicate detection. In *Proceedings of the International Workshop on Quality in Databases (QDB)*, 51–56.

Elmagarmid, A. K.; Ipeirotis, P. G.; and Verykios, V. S. 2007. Duplicate record detection: A survey. *Knowledge and Data Engineering, IEEE Transactions on* 19(1):1–16.

Ferraram, A.; Nikolov, A.; and Scharffe, F. 2013. Data linking for the semantic web. *Semantic Web: Ontology and Knowledge Base Enabled Tools, Services, and Applications* 169.

Hernández, M. A., and Stolfo, S. J. 1995. The merge/purge problem for large databases. In *ACM SIGMOD Record*, volume 24, 127–138. ACM.

Isele, R.; Jentzsch, A.; and Bizer, C. 2011. Efficient multidimensional blocking for link discovery without losing recall. In *WebDB*.

McCallum, A.; Nigam, K.; and Ungar, L. H. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 169–178. ACM.

Schmachtenberg, M.; Bizer, C.; and Paulheim, H. 2014. Adoption of the linked data best practices in different topical domains. In *The Semantic Web–ISWC 2014*. Springer. 245–260.