Privacy Preserving Transitive Record Linkage

Andrew Hill[1], Michael G. Kahn[2], Shaun Grannis[3,4], Chan Voong[2], Lisa Schilling[2], Toan C. Ong[2]
University of Colorado Denver[1], University of Colorado Anschutz Medical Campus[2],
Indiana University[3], Regenstrief Institute[4]

Introduction:

Record linkage (RL) is a process that determines which records belong to the same individual across datasets. [1] Linking patient records across multiple clinical data sources like clinical registries can help improve data quality and give a more complete picture of a patient's medical history. Clear-text RL algorithms use human-readable personally identifiable identifiers (PII) to determine linkages. Sharing PII increases risks to both data privacy and security. Privacy-Preserving Record Linkage (PPRL) is an alternative method that obfuscates all PII data before performing linkage, ensuring minimal risk of sensitive data being exposed. [2] Transitive record linkage captures linkages between multiple linked record pairs that may be missed using traditional pairwise only linkage methods. [3] For example, pairwise linkage may link Record A to Record B and Record B to Record C. Transitive record linkage adds a link between Record A and Record C. In this work, we developed and implemented a novel scalable method called Privacy Preserving Transitive Record Linkage (PPTRL) to generate both pairwise and transitive linkage results while maintaining the privacy and security of patient data.

Methods:

PPTRL has two steps: 1) performing traditional deterministic or probabilistic PPRL to generate pairwise linkages (Table 1a) and 2) using an undirected graph to generate transitive linkages. The strength of each pairwise linkage is measured by a confidence score values ranging from 0 (weakest) to 100 (strongest). Pairwise linkages are used as input to determine records which are linked together in a *record cluster*. Records in a record cluster are assigned with a common identifier called Network ID (Table 1b). A Network ID is an identifier used to refer to a single entity or individual. For example, consider the direct linkages (R1,R3) and (R2,R3) from Rows 1 and 2 in **Table 1a.** Records R1 and R2 both link to R3, but do not link to each other, since there is no direct (R1,R2) linkage in the table. However, records R1 and R2 are *indirectly* linked, since both connect to record R3. R1 and R2 are *transitively linked* and records R1, R2 and R3 are in the same record cluster. **Table 1b** represents the additional transitive linkage between R1 and R2 by assigning both records the same Network ID = 4001.

Table 1(a) pair-wise linkage; (b) transitive linkage.

#	Record ID 1	Record ID 2	Confidence
1	R1	R3	100
2	R2	R3	85
3	R4	R5	100

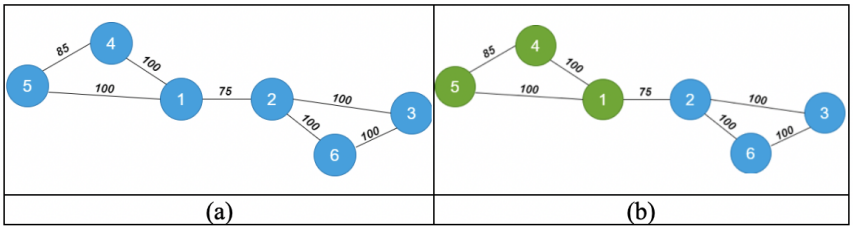
(a)

Record ID	Network ID
R1	4001
R2	4001
R3	4001
R4	4002
R5	4002

(b)

Transitive Linkage Computation: To calculate Network IDs from a pairwise linkage result, the direct (pairwise) linkage result (**Table 1a)** is formulated as an undirected graph *G*. Each record r_i is represented as a vertex in *G*, and each direct linkage between records r_1 and r_2 is represented as an edge $E = \{r_1, r_2\}$, Finding the Network IDs for each record is formulated as finding the maximally connected subgraphs, or connected components (*CC)* of *G*. A CC of *G* is a subgraph, such that every vertex in the *CC* is reachable by traversing edges from any other vertex. After partitioning graph *G* into a set of *CC*s, a unique Network ID is generated for each *CC* and assigned the same Network ID for all the nodes this subgraph.

Figure 1: Transitive linkage (a) before and (after) linkage refinement.



Transitive Linkage Refinement: Transitive Linkage identifies unique entities from a pairwise RL result. However, naively assuming that all connected components should be linked together may inadvertently generate false positive linkages resulting in records that belong to different individuals being assigned a common Network ID. To illustrate this error, consider a linkage scenario with two true entities $G_1 = \{1,4,5\}$ and $G_2 = \{2,3,6\}$. **Figure 1a** is a graphical representation of the linkage graph G described above. The number above each edge represents the pairwise linkage confidence. We can observe that entities G_1 and G_2 are both well connected within their own vertex groups but that both entities are connected by a linkage (1,2) having a weak confidence score = 75. Because we know that G_1 and G_2 are separate entities, the linkage (1,2) represents a false positive linkage. The Transitive Linkage algorithm will assign all records the same Network ID, since all records reside in the same CC. **Figure 1b** presents the correct output where the graph has been partitioned into two groups, represented by different colors. Each group will receive a separate Network ID. In order to avoid

problematic scenarios like **Figure 1a**, a greedy correlation clustering algorithm, first proposed in [3], modifies the Network IDs in each connected component of the linkage graph to reduce the effect of False Positive linkages. The algorithm begins with the Network IDs assigned by the original Transitive Linkage result as the initial set of cluster labels, which the algorithm then refines. To perform the refinement, the algorithm first assigns each CC a penalty score based on the similarities of disconnected vertices, and the dissimilarities of connected vertices. Then, the algorithm will attempt to find a cluster label assignment lower penalty score by changing the cluster labels of each vertex or by creating new cluster labels. Notably, each CC optimization is an independent operation, since each CC is disconnected from the rest of the graph. Thus, the refinement process can be executed in parallel across all CCs in a linkage result.

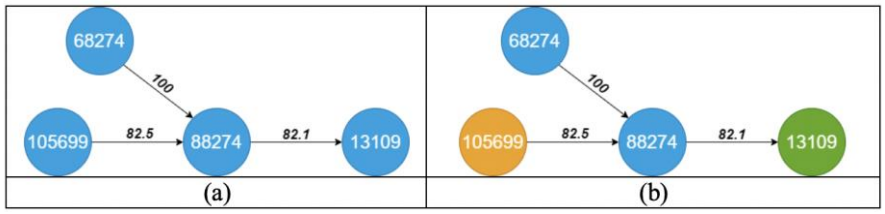
Results:

To test our transitive linkage method, we created a synthetic dataset with 115,000 records. We applied multiple corruption methods to simulate real-world data quality issues (missingness, typos, character transpositions, etc.). The data were linked using both probabilistic and deterministic PPRL methods. Network IDs were recorded for each record before and after the transitive linkage refinement process.

Table 2: Raw data using simulated corrupted records with known pairs.

id	gtid	first_name	last_name	address1	sex	city	zip	state	dob
13109	2	Artie	O'Noulane	7505 Gerald Pass	M	New York City	10019	New York	7/6/2017
68274	1	Artie	Guynemer	070 Dennis Drive	M	Tampa	33694	Florida	7/11/2017
88274	1	Artie	Guynemer	070 Dennis Drive	M	Tampa	33694	Florida	7/6/2017
105699	3	Artie	Boucher	710 Esker Place	M	San Antonio	78296	Texas	7/6/2017

Figure 2: (a) initial and (b) refined CC. Colors represent different assigned Network



The pairs of linkages which shared a Network ID prior to refinement but were assigned distinct Network IDs post-refinement (i.e. records which changed Network ID because of the refinement step) were recorded for analysis. Table 2 highlights one example with four records that were initially contained in the same CC. In the simulated data ground-truth labels (the gtid column in Table 2), Records 68724 and 88724 are known to belong to the same entity (gtid=1), while the remaining 2 records belong to separate, unrelated entities (gtid=2,3). Before refinement, all four records are linked transitively because of their shared relationship with Record 88274 as illustrated in Figure 2a. Records 88274 and 68274 are linked with a confidence of 100, indicating a perfect match. However, the (88274,105699) and (88274,13109) linkages are both false positives, caused by the similarity in the Name and date of birth (DOB) columns in Table 2. These linkages have lower confidence scores, indicating that a probabilistic method created the linkage. Following the refined step from Figure 2b, the algorithm splits both false positive records into separate Network IDs, (indicated by the color of the graph nodes. This cluster assignment was chosen because both false positive records are dissimilar from each other, since there is no (105699,13109) link. In addition, neither record shares a link with record 68274. Before refinement, transitive linkage result on the previously mentioned dataset produced 420 false positive linkages. After refinement, the transitive linkage result contained 200 false positives, showing that the algorithm removed over half of the false positive linkages found in the original transitive linkage result.

Conclusion:

PPTRL adds additional true positive linkages to the traditional linkage results by including both direct and indirect linkages. After the initial graph partitioning, the PPTRL result is refined by a parallel greedy clustering algorithm to reduce the occurrence of false positive linkages.

References:

1 Herzog, Thomas & Scheuren, Fritz & Winkler, William. (2007). Data Quality and Record Linkage. 10.1007/0-387-69505-2.

2. Verykios, Vassilios & Christen, Peter. (2013). Privacy-preserving record linkage. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 3. 10.1002/widm.1101.

3. Anja Gruenheid, Xin Luna Dong, and Divesh Srivastava. 2014. Incremental record linkage. Proc. VLDB Endow. 7, 9 (May 2014), 697–708. DOI:https://doi.org/10.14778/2732939.2732943