

# Discovering Similarity Inclusion Dependencies

Extended Abstract

Youri Kaminsky<sup>1</sup>, Eduardo H. M. Pena<sup>2</sup> and Felix Naumann<sup>1</sup>

<sup>1</sup>Hasso Plattner Institute, University of Potsdam, Germany

<sup>2</sup>Federal University of Technology – Paraná, Brazil

## Abstract

In this extended abstract<sup>1</sup>, we introduce similarity inclusion dependencies (sINDs). They extend the traditional notion of inclusion dependencies (INDs) by relaxing the dependency definition using a similarity measure: otherwise valid INDs can be discovered even in the presence of minor data errors, such as typos or different formatting. We present SAWFISH, the first algorithm to discover all sINDs in a given dataset efficiently. Our algorithm combines approaches for the discovery of traditional INDs and string similarity joins with a novel sliding-window approach and lazy candidate validation.

## 1. Similarity Inclusion Dependencies

Data dependencies are an important type of metadata and are a crucial component of data profiling. Traditional inclusion dependencies (INDs) express that the tuples of one column-combination are contained in the tuples of another column-combination. Their discovery assumes clean data: *all* tuples of the dependent column-combination must be exactly *equal* in the referenced column-combination. However, the ever-increasing volume of data also leads to more “dirty” data [2], giving rise to relaxed dependencies [3]. While there has been research for other dependencies to allow for similar values, such as functional dependencies [4], there is no corresponding notion yet for INDs. Thus, we introduce *similarity inclusion dependencies* (sINDs). In contrast to traditional INDs, an sIND holds if, for all dependent values, there exists a referenced value that is at least *similar*. sINDs support arbitrary similarity measures and configurable similarity thresholds. In this work, we consider representatives of both edit-based and token-based similarity measures: the Levenshtein distance [5] and the Jaccard similarity.

sINDs can serve many of the same use-cases as traditional INDs, including, in particular, the discovery of foreign key candidates and joinable partners [6], as well as assisting in schema design [7]. To illustrate the usefulness of sINDs, we present an example for a fictitious football tournament in Figure 1. Table (a) shows the final results after all games, and Table (b) lists all goalkeepers in the tournament. We would assume that the values of the *club* column of Table (b) are contained in the values of the *name* column of Table (a). However, multiple goalkeepers


---


<sup>1</sup>Condensed version of a research paper at SIGMOD 2023 [1]

LWDA’24: *Lernen, Wissen, Daten, Analysen*. September 23–25, 2024, Würzburg, Germany

✉ youri.kaminsky@hpi.de (Y. Kaminsky); eduardopena@utfpr.edu.br (E. H. M. Pena); felix.naumann@hpi.de (F. Naumann)

ORCID 0009-0007-6547-592X (Y. Kaminsky); 0000-0002-4852-3113 (E. H. M. Pena); 0000-0002-4483-1389 (F. Naumann)

 © 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

made minor spelling mistakes in their club name. Despite these mistakes, we want to discover the dependency.

**Figure 1:** Example relations of a football tournament

(a) Final Results		(b) Participating Goalkeepers	
<b>results</b>		<b>goalie</b>	
name	points	p_id	club
SpVgg Beelitz	4	202	SpVgg Beelitz
Potsdamer SC	9	216	SpVgg Beelittz
SpVgg Bernau	4	469	Potsdamer SC
VfL Potsdam	0	617	SpVgg Be_nau
		692	VjL Potsdam
		853	Potsdamer SC

## 2. The SAWFISH Algorithm and Beyond

The general validation idea of SAWFISH is that it is sufficient to find a single counterexample to invalidate an sIND candidate. Since it is infeasible to directly compare every dependent value to every referenced value, SAWFISH uses an inverted index. After building the index for a referenced column, each dependent column is probed against the index. If each dependent value is similar to at least one referenced value, the sIND is emitted as a valid dependency. Given that a dependent value can only match index values within a certain interval, we can reduce the size of the index. Like a sliding window through the occurring lengths of the dataset, we iterate the dataset length-wise and build new indices only on-demand while removing unused indices. Besides reducing the index size, we also do not need to build the entire index if a column is no longer referenced by any other column.

In the original evaluation, we show that SAWFISH scales well in the number of rows, and in the number of columns. Compared to a naïve string similarity join baseline, we outperform it by a factor of up to 6.5. To assess whether the discovered similarity inclusion dependencies can indeed indicate joinability in the presence of typos or other data errors, we ran SAWFISH on a subset of the *2015 Web Table Corpus (WTC)* [8]. In total, we observe 1044 sINDs that are not traditional INDs. We manually annotated them to assess their genuineness. Overall, we found that there are 161 (15%) meaningful sINDs, 671 (64%) coincidental sINDs, and 212 (20%) erroneous (wrong header detection) sINDs. However, we find two criteria for the WTC data that reduce the number of coincidental sINDs significantly. First, the maximum number of characters of any value of the dependent side is above three. Second, the portion of dependent values that match only similarly to a value on the referenced side is below 30%. Given these two filters, there are only 33 (20%) coincidental sINDs, but 101 (64%) meaningful sINDs.

An obvious next step, after discovering sINDs, is to “repair” them by slightly modifying dependent or referenced values to achieve proper inclusion. In future work, we address the challenge of identifying *minimal* repairs under various distance measures.

## References

- [1] Y. Kaminsky, E. H. M. Pena, F. Naumann, Discovering similarity inclusion dependencies, *Proceedings of the ACM on Management of Data (PACMMOD)* 1 (2023). doi:10.1145/3588929.
- [2] R. Marsh, Drowning in dirty data? it's time to sink or swim: A four-stage methodology for total data quality management, *Journal of Database Marketing and Customer Strategy Management* 12 (2005) 105–112. doi:10.1057/palgrave.dbm.3240247.
- [3] L. Caruccio, V. Deufemia, G. Polese, Relaxed functional dependencies – a survey of approaches, *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 28 (2016) 147–165. doi:10.1109/TKDE.2015.2472010.
- [4] P. Schirmer, T. Papenbrock, I. Koumarelas, F. Naumann, Efficient discovery of matching dependencies, *ACM Transactions on Database Systems (TODS)* 45 (2020). doi:10.1145/3392778.
- [5] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, in: *Soviet Physics Doklady*, volume 10, 1966, pp. 707–710.
- [6] R. J. Miller, M. A. Hernández, L. M. Haas, L. Yan, C. T. Howard Ho, R. Fagin, L. Popa, The clio project: Managing heterogeneity, *SIGMOD Rec.* 30 (2001) 78–83. doi:10.1145/373626.373713.
- [7] M. Levene, M. W. Vincent, Justification for inclusion dependency normal form, *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 12 (2000) 281–291. doi:10.1109/69.842267.
- [8] O. Lehmborg, D. Ritze, R. Meusel, C. Bizer, A large public corpus of web tables containing time and context metadata, in: *Proceedings of the International Conference Companion on World Wide Web*, International World Wide Web Conferences Steering Committee, 2016, p. 75–76. doi:10.1145/2872518.2889386.