8 Parameters of the Algorithm

The use of Jaccard index to assess similarity between addresses in our algorithm is optional. Our implicit assumption is that there exists a function $d(x, y)$ which assesses the similarity between two addresses x and y. Blocking can reduce the number of evaluations of $d(x, y)$ without missing links, if $d(x, y) > \tau$ indicating x and y share a common token. In our two-round linkage, our implicit function is

$$d(x,y) = \begin{cases} J_{char}(x,y) & \text{if } J_{phrase}(x,y) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

One may design any other implicit function instead, replacing the Jaccard index with any other measurement.

The Jaccard index in round 2 of the comparison can also be replaced by almost any other similarity function, for example the Monge-Elkan function [13], which is suitable for addresses.

9 Conclusion

We have presented in this paper a novel address-linkage algorithm that:

1. links addresses as free text;
2. uses data-driven blocking keys;
3. extends the inverted index data structure to facilitate large-scale address linking;
4. is robust against data-quality issues; and
5. is practical and scalable.

The simplicity of the solution - a great virtue in large-scale industrial applications - may belie the slightly tortuous journey leading to its discovery; a journey laden with the corpses of a wide-range of seemingly good ideas like compressive sensing and other matrix factorisation and dimensionality-reduction techniques, nearest-neighbour algorithms like KD-trees, ElasticSearch with custom rescoring functions [8], rules-based expert systems, and implementation languages that range from low-level C, to R, Python, SQL and more. In retrospect, our algorithm can be interpreted as an application of a signature-based approach to efficiently compute set-similarity joins [1], where the abstract concept of sets is replaced with carefully considered set-representations of addresses, with a modern twist in its implementation on state-of-the-art parallel databases to lift the algorithm's scalability to potentially petabyte-sized datasets.

References

1. Arasu, A., Ganti, V., Kaushik, R.: Efficient exact set-similarity joins. In: Proceedings of the 32nd International Conference on Very Large Data Bases. pp. 918–929. VLDB Endowment (2006)