

# Fast-ER: GPU-Accelerated Record Linkage and Deduplication in Python

Jacob Morrier<sup>1</sup>, Sulekha Kishore<sup>1</sup>, and R. Michael Alvarez<sup>1</sup>

<sup>1</sup> Division of the Humanities and Social Sciences, California Institute of Technology, USA

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#)
- [Repository](#)
- [Archive](#)

Editor: [Open Journals](#)

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

## Summary

Record linkage, or “entity resolution,” consists of identifying matching records across multiple datasets that lack common unique identifiers. Deduplication, on the other hand, involves recognizing duplicate entries within a dataset in which unique identifiers are either inconsistent or missing. These methods are essential tools for research in fields like social and health sciences, among others (e.g., [Jutte et al., 2011](#); [Kim et al., 2020](#); [Kwiek & Roszka, 2021](#); [Ruggles et al., 2018](#); [Yoder, 2020](#)).

Both tasks often require calculating string similarity metrics, such as the Jaro-Winkler metric, for all pairs of values between datasets. The Fast-ER package harnesses the computational power of graphical processing units (GPUs) to accelerate this dramatically. It estimates the widely used Fellegi-Sunter probabilistic model and performs the computationally intensive preprocessing steps, including calculating string similarity metrics, on CUDA-enabled GPUs.

Fast-ER runs over 35 times faster than the leading CPU-powered software implementation, reducing execution time from hours to minutes. This significantly enhances the scalability of record linkage and deduplication for large datasets.

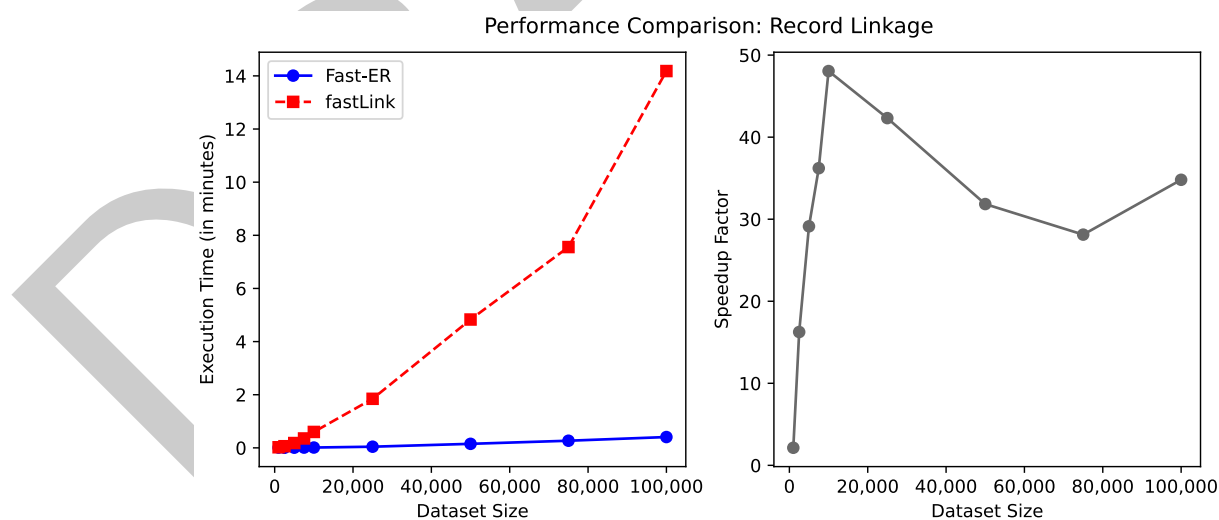
## Statement of Need

Record linkage and deduplication typically involve calculating string similarity metrics, such as the Jaro-Winkler metric ([Winkler, 1990](#)), for all pairs of values between two datasets or within a dataset. Although these calculations are simple, the number of comparisons grows exponentially with the number of observations. For instance, when linking observations from two datasets, each with 1,000,000 observations, adding just one more observation to either dataset results in an additional 1,000,000 comparisons. This makes record linkage and deduplication prohibitively expensive to perform, even for datasets of moderate size.

GPUs were developed in the 1970s to accelerate digital image processing. Unlike central processing units (CPUs), designed for the sequential execution of a single thread of instructions with minimal latency, GPUs are optimized for performing hundreds of operations simultaneously ([Kirk & Hwu, 2017](#)). Early applications focused on geometric transformations, such as rotating and translating vertices between coordinate systems, and texture mapping. GPUs can also be used for non-graphical computations. They are especially well-suited for high-throughput computations that can be broken down into identical, independent calculations, such as those exhibiting data parallelism, in which the same instructions are executed individually over many data points. This stems from GPUs’ Single Instruction, Multiple Data (SIMD) architecture. Concretely, the shader pipelines of modern GPUs can execute “compute kernels,” analogous to instructions in a “for loop.” However, rather than running sequentially, these operations are executed simultaneously across inputs. As a result, GPUs can often deliver performance orders of magnitude faster than traditional CPUs.

Our GPU-accelerated implementation of record linkage and deduplication relies heavily on the CuPy library, an open-source library for array-based numerical computations on GPUs in Python (Okuta et al., 2017). Built on NVIDIA's CUDA parallel computing model, CuPy has an intuitive application programming interface (API) that closely mirrors that of NumPy. This makes it a natural solution for Python developers who want to harness the immense computational power of GPUs.

The primary challenge in calculating the Jaro-Winkler similarity metric, and more generally in handling strings, on GPUs stems from the fact that the latter do not natively support jagged arrays, also called "arrays of arrays." A string is an array of characters, so an array of strings is, in effect, an array of arrays of characters. This limitation similarly applies to "arrays of arrays" for other data types. A simple solution is to convert jagged arrays into a different data structure: the Arrow columnar format (Apache Software Foundation, 2024). Numerous libraries have adopted this format, including PyArrow and RAPIDS cuDF (RAPIDS Development Team, 2023). In short, this approach consists of storing jagged arrays in a primitive layout, that is, a long array of contiguous values of the same data type and fixed memory size (e.g., a long array of characters), paired with a sequence of indices that indicate the starting position of each inner array within the outer array. Concretely, with this approach, arrays of strings are flattened into a single array of characters. The character array and its index buffers can be efficiently stored and manipulated on GPUs. For example, the array of strings ['David', 'Elizabeth', 'James', 'Jennifer', 'John', 'Linda', 'Mary', 'Michael', 'Patricia', 'Robert'] can be represented as an array of characters ['D', 'a', 'v', 'i', 'd', 'E', 'l', 'i', 'z', 'a', 'b', 'e', 't', 'h', 'J', 'a', 'm', 'e', 's', 'J', 'e', 'n', 'n', 'i', 'f', 'e', 'r', 'J', 'o', 'h', 'n', 'L', 'i', 'n', 'd', 'a', 'M', 'a', 'r', 'y', 'M', 'i', 'c', 'h', 'a', 'e', 'l', 'P', 'a', 't', 'r', 'i', 'c', 'i', 'a', 'R', 'o', 'b', 'e', 'r', 't'], along with the following sequence of indices, [0, 5, 14, 19, 27, 31, 36, 40, 47, 55]. This strategy is efficient in terms of access patterns and memory usage.



**Figure 1:** Performance Comparison Between Fast-ER and fastLink for Record Linkage

To illustrate the performance of GPU-accelerated record linkage, we compare the performance of our library with that of the leading CPU-powered software implementation, fastLink (Enamorado et al., 2017, 2019). We join two extracts of North Carolina voter registration rolls of varying sizes (from 1,000 to 100,000 observations), comparing first names, last names, house numbers, and street names for fuzzy matching and birth years for exact matching. The datasets have 50% overlapping records. We injected noise into 5% of the records through various transformations: character addition, character deletion, random shuffling of values,

replacing a character with another, and swapping two adjacent characters. The results in Figure 1 confirm that our GPU-accelerated implementation delivers speeds over 35 times faster than fastLink.

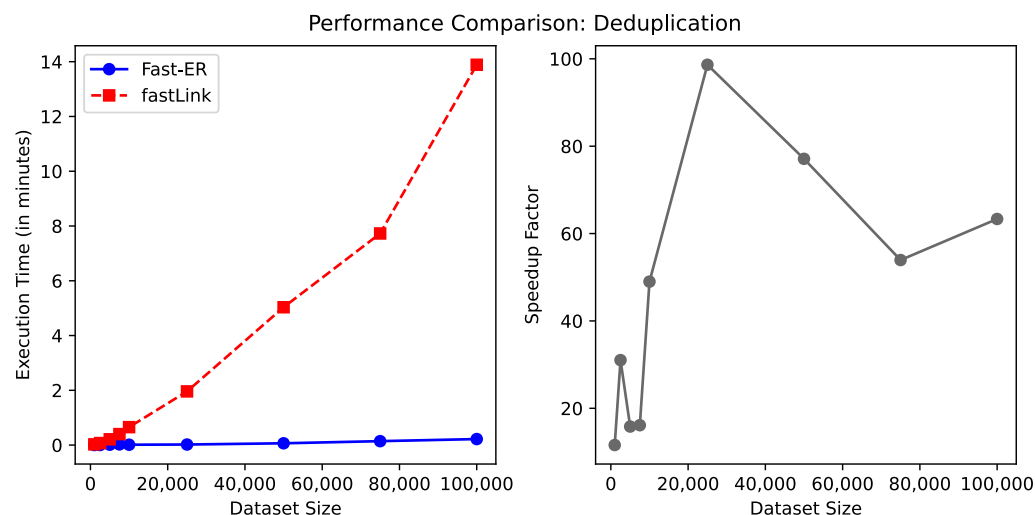


Figure 2: Performance Comparison Between Fast-ER and fastLink for Deduplication

Analogously, we compare the performance of our library for deduplication with that of the leading CPU-powered software implementation. Deduplication was executed on one of the datasets described above. The results in Figure 2 confirm that our GPU-accelerated implementation runs over 60 times faster than fastLink.

## References

- Apache Software Foundation. (2024). *Arrow Columnar Format*. <https://arrow.apache.org/docs/format/Columnar.html>
- Enamorado, T., Fifield, B., & Imai, K. (2017). *fastLink: Fast Probabilistic Record Linkage with Missing Data*. <https://github.com/kosukeimai/fastLink>
- Enamorado, T., Fifield, B., & Imai, K. (2019). Using a Probabilistic Model to Assist Merging of Large-Scale Administrative Records. *American Political Science Review*, 113(2), 353–371. <https://doi.org/10.1017/S0003055418000783>
- Jutte, D. P., Roos, L. L., & Brownell, M. D. (2011). Administrative Record Linkage as a Tool for Public Health Research. *Annual Review of Public Health*, 32, 91–108. <https://doi.org/10.1146/annurev-publhealth-031210-100700>
- Kim, S. S., Schneider, S., & Alvarez, R. M. (2020). Evaluating the Quality of Changes in Voter Registration Databases: Part of Special Symposium on Election Sciences. *American Politics Research*, 48(6), 670–676. <https://doi.org/10.1177/1532673X19870512>
- Kirk, D. B., & Hwu, W. W. H. (2017). *Programming Massively Parallel Processors: A Hands-on Approach* (Third Edition). Morgan Kaufmann.
- Kwiek, M., & Roszka, W. (2021). Gender disparities in international research collaboration: A study of 25,000 university professors. *Journal of Economic Surveys*, 35(5), 1344–1380. <https://doi.org/10.1111/joes.12395>
- Okuta, R., Unno, Y., Nishino, D., Hido, S., & Loomis, C. (2017). CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations. *Proceedings of Workshop on Machine*

- 103     *Learning Systems (LearningSys) in the Thirty-First Annual Conference on Neural Information*  
104     *Processing Systems (NIPS)*. [http://learningsys.org/nips17/assets/papers/paper\\_16.pdf](http://learningsys.org/nips17/assets/papers/paper_16.pdf)
- 105     RAPIDS Development Team. (2023). *RAPIDS: Libraries for End to End GPU Data Science*.  
106     <https://rapids.ai>
- 107     Ruggles, S., Fitch, C. A., & Roberts, E. (2018). Historical Census Record Linkage. *Annual*  
108     *Review of Sociology*, 44, 19–37. <https://doi.org/10.1146/annurev-soc-073117-041447>
- 109     Winkler, W. E. (1990). String Comparator Metrics and Enhanced Decision Rules in the  
110     Fellegi-Sunter Model of Record Linkage. *Proceedings of the Section on Survey Research*  
111     *Methods*, 354–359.
- 112     Yoder, J. (2020). Does Property Ownership Lead to Participation in Local Politics? Evidence  
113     from Property Records and Meeting Minutes. *American Political Science Review*, 114(4),  
114     1213–1229. <https://doi.org/10.1017/S0003055420000556>

DRAFT