The Journal of Open Source Software

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

Jacob Morrier[1], **Sulekha Kishore**[1], **and R. Michael Alvarez**[1]

1 Division of the Humanities and Social Sciences, California Institute of Technology, USA

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Record linkage, also called "entity resolution," consists of identifying matching records across different datasets, even when no consistent common identifiers are available. Deduplication, on the other hand, consists of identifying duplicate entries within a dataset when consistent unique identifiers are inconsistent or missing. Both tasks typically involve computing string similarity metrics, such as the Jaro-Winkler metric, for all pairs of values between the datasets.

The `Fast-ER` package harnesses the computational power of graphical processing units (GPUs) to dramatically accelerate these processes. It estimates the widely used Fellegi-Sunter model and performs the computationally intensive preprocessing steps, including the calculation of string similarity metrics, on CUDA-enabled GPUs.

`Fast-ER` runs over 35 times faster than the leading CPU-powered software implementation, reducing processing 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. 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 first 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, where the same instructions are applied individually to 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 they 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.

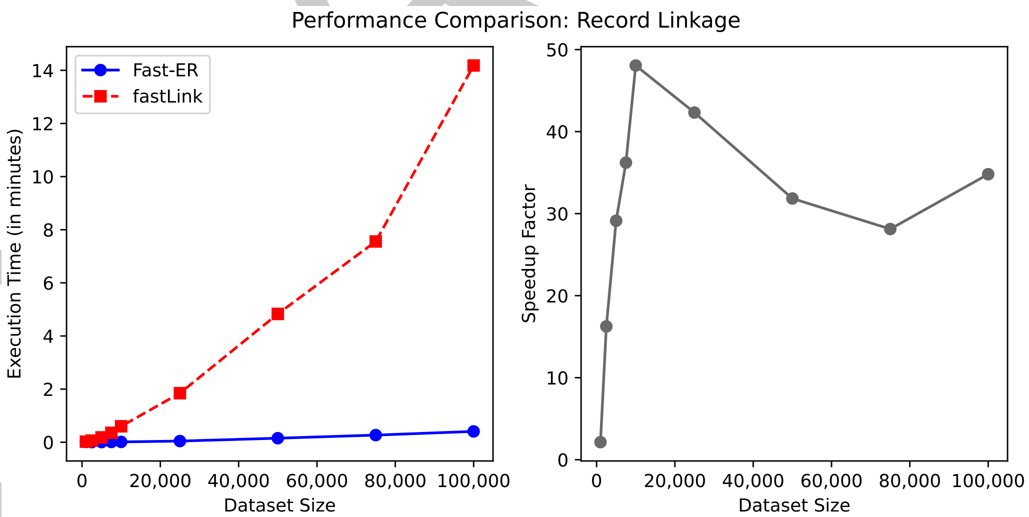

Figure 1: Performance Comparison Between `Fast-ER` and `fastLink` for Record Linkage

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 excerpts 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 confirm that our GPU-accelerated implementation is consistently faster than `fastLink`, delivering speed improvements exceeding 35 times.

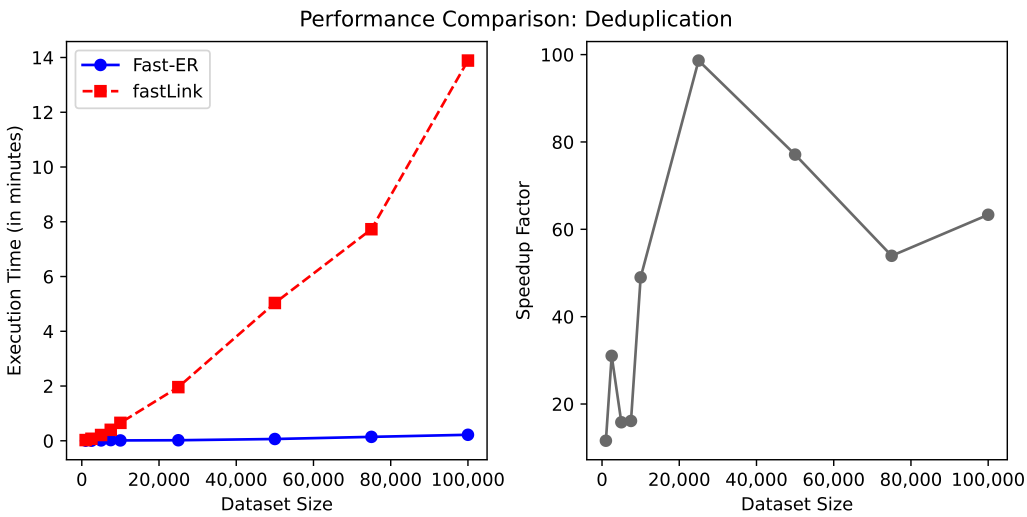

Figure 2: Performance Comparison Between `Fast-ER` and `fastLink` for Deduplication

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 confirm that our GPU-accelerated implementation is consistently faster than `fastLink`, delivering speed improvements exceeding 60 times.

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. In *GitHub*. 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

Kirk, D. B., & Hwu, W. W. H. (2017). *Programming Massively Parallel Processors: A Hands-on Approach* (Third). Morgan Kaufmann.

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 Learning Systems (LearningSys) in the Thirty-First Annual Conference on Neural Information Processing Systems (NIPS)*. http://learningsys.org/nips17/assets/papers/paper_16.pdf

RAPIDS Development Team. (2023). *RAPIDS: Libraries for End to End GPU Data Science*. https://rapids.ai

Winkler, W. E. (1990). String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. *Proceedings of the Section on Survey Research Methods*, 354–359.