privJedAI: A Unified Open-Source Pipeline for Privacy-Preserving Record Linkage

Lefteris Stetsikas
National and Kapodistrian University Of Athens
Athens, Greece
istetsikas@di.uoa.gr

Dimitrios Karapiperis
International Hellenic University
Thessaloniki, Greece
dkarapiperis@ihu.edu.gr

George Papadakis
National and Kapodistrian University of Athens
Athens, Greece
gpapadis@di.uoa.gr

Manolis Koubarakis
National and Kapodistrian University of Athens
Athens, Greece
koubarak@di.uoa.gr

ABSTRACT

Privacy-Preserving Record Linkage (PPRL) is a crucial task for interlinking different, but overlapping data sources without compromising their privacy. PPRL has evolved from simple hash-based exact matching methods to more sophisticated ones in the last two decades. However, existing open-source PPRL systems have limited utility, as they implement very few methods, typically of their creators, and thus, they do not facilitate users to build and benchmark different end-to-end pipelines. To address this shortcoming, this work introduces `privJedAI`, a new open-source library for PPRL that implements the Filtering-Verification framework, offering a plethora of state-of-the-art methods per workflow step. privJedAI leverages Python's data ecosystem to allow users to combine state-of-the-art techniques into end-to-end PPRL pipelines in an intuitive way that accommodates researchers and practitioners of any expertise. Special care is also taken to facilitate the benchmarking of the resulting pipelines to assess their trade-off between effectiveness and time efficiency across a series of established or custom datasets. We present the architecture of privJedAI, its unique features and explain how it can be used in practice.

PVLDB Reference Format:
Lefteris Stetsikas, Dimitrios Karapiperis, George Papadakis, and Manolis Koubarakis. privJedAI: A Unified Open-Source Pipeline for Privacy-Preserving Record Linkage. PVLDB, 19(12): 4806 - 4809, 2026.
doi:10.14778/3827998.3828127

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/AI-team-UoA/privJedAI.

1 INTRODUCTION

Privacy-Preserving Record Linkage (PPRL) is a critical data integration technique that enables the matching of records referring to the same entity across disparate databases without exposing Personally Identifiable Information (PII) [3]. In the simplest form, PPRL involves two data owners with individually clean, but overlapping datasets, D_1 and D_2, that want to interlink them such that no party learns any additional information about other datasets beyond what can be inferred from the detected matches and its own data. By employing encryption, hashing, and other obfuscation methods, PPRL allows persons and organizations to combine data from multiple sources, while maintaining strict compliance with privacy regulations such as HIPAA and GDPR, and adhering to FAIR principles (Findable, Accessible, Interoperable, and Reusable).

The literature on PPRL is extensive, with the proposed solutions spanning a wide spectrum that ranges from hash-based methods [14] and secure multi-party computation protocols [9, 11] to differential privacy techniques [10] and Bloom filter encodings [1, 15, 17]. These works address the main challenges of PPRL:

(1) *Scalability*. The brute-force approach to PPRL compares every entity from one data source (D_1) with all entities from the other data sources (D_2). This corresponds to a quadratic time complexity that cannot scale to large data sources.
(2) *Effectiveness*. Real-word data are noisy, incomplete, and inconsistent, yet a successful PPRL method should avoid false matches without missing true matches.
(3) *Resilience*. PPRL solutions should protect the privacy of the original data, avoiding information leakage through adversarial attacks like dictionary, frequency and collusion attacks.

Various open-source tools that are specialized in PPRL, such as SOEMPI [13], MainSEL [12], PRIMAT [2], Anonlink, LSHDB [4] and AMPPERE [16]. All of these tools follow the *Encoding-Filtering-Verification* framework in Figure 1: First, an encoding technique is applied to encrypt the input data, thus avoiding information leakage. This is typically achieved by mapping sensitive quasi-identifiers, such as names or addresses, into secure, distance-preserving structures, such as Bloom filters, which allow the system to evaluate record similarity without ever exposing the raw plaintext values. Next, blocking (a.k.a. Filtering) is applied to reduce the computational cost from the Cartesian product to the most similar candidate pairs. These pairs form the candidate set C, such that $|C| \ll |D_1| \times |D_2|$. Finally, matching (a.k.a. Verification) is applied to each pair in C so as to distinguish the highly similar, but non-matching entities from the real duplicates.

However, these tools have the following drawbacks:

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828127

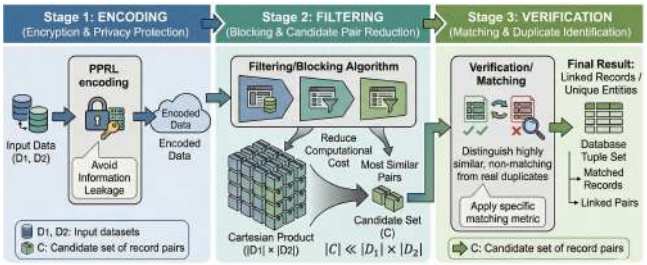

Figure 1: PPRL's Encoding-Filtering-Verification framework.

(1) They offer a limited variety of methods for each step of the Encoding-Filtering-Verification framework. In fact, they disregard the state-of-the-art techniques from the literature, typically building pipelines that exclusively incorporate the methods of their own creators.
(2) They disregard the literature of each PPRL workflow step. Especially, for Filtering and Verification, numerous methods have been developed for the generic Record Linkage task, but could be applied to PPRL, too, with the appropriate adaptations.
(3) They suffer from limited scalability due to their reliance on exhaustive comparisons or standard blocking. They overlook the integration of Approximate Nearest Neighbor Search (ANNS) indexes, disregarding a highly efficient method for retrieving candidate matches in large-scale scenarios.
(4) They treat Matching as a binary classification tasks, return its output as the final result. This way, though, they disregard the 1-1 constraint that dictates that every entity from D_1 matches with at most one entity from D_2 (and vice versa), because they are individually clean (i.e., duplicate-free).
(5) They do not facilitate the experimental comparison between different end-to-end pipelines in order to identify the top performing one(s) for the data at hand.

To address these shortcomings, we introduce `privJedAI`, a new, comprehensive open-source Python library that unifies the state-of-the-art PPRL and RL methods into end-to-end pipelines. More specifically, `privJedAI` offers the following unique characteristics:

(1) It relies on a *modular architecture* that facilitates its extension with individual methods in each step of the Encoding-Filtering-Verification framework. The only requirement is that they implement the same interface, which specifies the input and the output of the corresponding workflow step. Thus, `privJedAI` acts as an *extensible library* of the state-of-the-art techniques for each step of the Encoding-Filtering-Verification framework.
(2) `privJedAI` captures a large part of the literature for each PPRL workflow step, i.e., Encoding, Filtering and Verification. Especially for Filtering, `privJedAI` incorporates the latest advances in generic Record Linkage, such as meta-blocking methods and techniques for ANNS [6].
(3) `privJedAI` enhances the Verification step too, by combining the matching algorithms with clustering techniques. Specifically, all other PPRL tools may contain conflicts in their end results, such as $e_i \in D_1$ matches with $e_j \in D_2$ and with $e_l \in D_2$. In contrast, `privJedAI` leverages the 1-1 constraint between the input datasets, which dictates that each entity from D_1 is matching with at most another entity form D_2, and vice versa: it converts the output of matching algorithms into a bipartite graph, with conflicts resolved with one of the state-of-the-art bipartite graph matching algorithms for Record Linkage [8].
(4) `privJedAI` enables users to easily build and experiment with thousands of different end-to-end pipelines. This is facilitated through tutorials and through the Documentation component, which specifies for each method the configuration parameters, their default values as well as their domain. This way, `privJedAI` accommodates both expert and novice users, as long as they are familiar with the Python ecosystem.

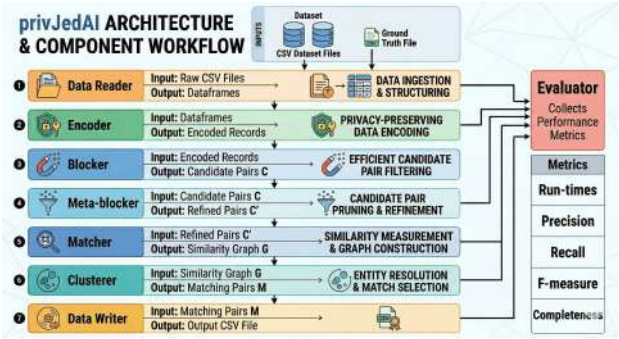

Figure 2: Overview of `privJedAI`'s end-to-end workflow, with each step corresponding to a different component.

(4) `privJedAI` enables users to easily build and experiment with thousands of different end-to-end pipelines. This is facilitated through tutorials and through the Documentation component, which specifies for each method the configuration parameters, their default values as well as their domain. This way, `privJedAI` accommodates both expert and novice users, as long as they are familiar with the Python ecosystem.

Below, we present the architecture of `privJedAI` (Sec. 2) and its supported methods (Sec. 3), we compare it with the other open-source PPRL systems (Sec. 4) and present the demo scenario (Sec. 5).

2 PRIVJEDAI ARCHITECTURE

The `privJedAI` system consists of the components illustrated in Figure 2. Each component corresponds to a workflow step of the end-to-end PPRL pipeline and implements its own interface, which specifies the input and output of the respective methods. This way, `privJedAI` can be seamlessly extended with new techniques in every workflow step, as long as they implement the same interface.

We now briefly discuss each component:

- **Data Reader.** It receives as input datasets in the form of CSV files, which are loaded from the disk. The users can specify which quasi-identifiers and features will be utilized throughout the PPRL pipeline. Ground truth data can also be ingested at this stage to facilitate performance evaluation. Its output is a Pandas dataframe for each file along with an optional one for the groundtruth.
- **Encoder.** This component receives as input the Pandas dataframes of the input datasets. `privJedAI` enables data owners to encode their datasets locally to protect PII. To this end, it provides multiple encoding techniques, offering the flexibility to hash at the level of whole strings, tokens, q-grams, multiple q-grams, skip-grams, Metaphone phonetic codes, and Bloom filter generation. The output is a data object containing the encoded values for each defined attribute, which is then saved to disk to be shared with the Trusted Third Party. If no attribute is defined, cryptographic long-term keys (CLKs) are generated for each Record.
- **Blocker.** This component conveys methods that restrict the computational cost to highly similar records, clustering them into a set of candidate pairs C. Therefore, this component receives as

input the output of the `Encoder` and produces as output a set of candidate pairs C in the form of a Python dictionary, where the keys are the indices from the first dataset and each value is a set of indices from the second dataset. Apart from the traditional LSH techniques, `privJedAI` implements ANNS methods based on FAISS that use Bloom filters as vectors.

- **Meta-blocker.** It receives as input a set of candidate pairs C generated by the Blocker and returns as output a refined set C' that contains no redundant pairs and a much lower number of superfluous ones, while retaining most true duplicates.
- **Matcher.** The input to this component is a set of candidate pairs C, while its output is a bipartite graph $\mathcal{G}$.
- **Clusterer.** This component receives as input the bipartite graph $\mathcal{G}$ generated by the Matcher and returns as output the detected set of matching pairs $\mathcal{M}$.
- **Evaluator.** This component receives as input an end-to-end pipeline and returns as output a dataframe with the performance of the intermediate and the final results. For all steps, it reports run-times, while the effectiveness measures differ per workflow step: for Matcher and Clusterer, it reports precision, recall and F-Measure, whereas for Blocker and Meta-blocker, it reports pairs completeness, pairs quality and number of candidate pairs.
- **Data Writer.** It receives as input the detected matching pairs $\mathcal{M}$ and stores them on the specified file on the disk.

Note that privJedAi also includes the **Documentation** component, which contains textual explanations about each workflow step, each supported method and its configuration parameters, along with their default values and domain.

3 SUPPORTED METHODS

`privJedAI` is an open-source system that unifies all other open-source PPRL libraries into a modular, user-friendly framework: PPRL-toolkit[1], Linkja[2], PRIMAT[3] [2], AMPPERE[4] [16], and Anonlink[5]. As a result, `privJedAI` offers in each step multiple competitive approaches, with the users building end-to-end pipelines by selecting one method per workflow step. More specifically, the following methods are supported per workflow step:

- **Encoding and Hashing:** For every feature within a record, `privJedAI` generates a Bloom Filter, which is a highly memory-efficient, probabilistic data structure that enables rapid membership queries. These filters are populated by hashing the textual data, with users choosing among the following hashing configurations: (i) *Salted-String*. The entire string is hashed and stored in a Bloom Filter. Salting permits the same data to be hashed multiple times for storage in distinct spaces. (ii) *Salted-QGrams*. Textual data is divided into q-grams before hashing and insertion. Like the string method, salting enables multi-space storage. (iii) *Salted-SkipQGrams*. Textual data is divided into q-grams with intermittent skipping. These skip-grams are then hashed and stored, with salting available for diverse storage mapping. (iv) *Salted-Tokens*. The text is split into discrete tokens, hashed, and stored. Salting creates multiple independent hash spaces.
- **Blocking:** This phase is critical for efficiently reducing the massive candidate pair space. To achieve this, `privJedAI` offers both probabilistic and index-based filtering methods. First, it implements two variants of Hamming Locality-Sensitive Hashing (LSH), adapted from the PRIMAT and Anonlink toolkits, which probabilistically group highly similar encoded records into shared buckets. Second, `privJedAI` leverages ANNS on the binary encodings to rapidly retrieve the K closest neighbors and form high-quality candidate pairs. This is powered by the FAISS[6] library, utilizing the binary HNSW index [5] and Multi-Index Hashing [7]. Finally, an exact binary flat index is included for brute-force Exact Nearest Neighbor search, serving as a gold-standard baseline for evaluating the approximate methods.
- **Meta-Blocking:** `privJedAI` offers the following advanced Meta-Blocking techniques to further refine the candidate set [6]: (i) *Weighted Edge Pruning* retains comparisons with a weight exceeding the average edge weight in the blocking graph. (ii) *Cardinality Edge Pruning* retains comparisons corresponding to the top-K weighted edges globally within the graph. (iii) *Weighted Node Pruning* retains for every entity comparisons corresponding to edges that exceed the average edge weight in the respective node's neighborhood.Cardinality (iv) *Node Pruning* retains For every entity comparisons corresponding to its top-K weighted edges. (v) *BLAST* retains comparisons where the edge weight exceeds one-quarter of the sum of the maximum edge weights in the two adjacent node neighborhoods. (vi) *Reciprocal Cardinality Node Pruning* retains comparisons whose corresponding edges are among the top-K weighted for both adjacent nodes. (vii) *Reciprocal Weighted Node Pruning* retains comparisons where the edge weight exceeds the average weight in both adjacent node neighborhoods.
- **Matching:** `privJedAI` calculates a similarity score within the interval $[0, 1]$ for each candidate pair to quantify the likelihood of a match. This is achieved by comparing the Bloom filters and computing the average similarity. Available similarity metrics include Dice, Jaccard, Cosine, and Soft Cosine Measures.
- **Clustering:** To eliminate matching conflicts based on the 1-1 constraint, (i.e., the fact that two distinct datasets are duplicate-free), `privJedAI` employs advanced bipartite graph matching algorithms [8]. Because entity matching graphs frequently contain conflicting edges, which link multiple nodes from D_1 to the same node from the D_2 or vice versa, clustering resolves these by associating each entity from D_1 with the most likely match from D_2. The implemented algorithms include Connected Components, Unique Mapping, Kiraly, and Center Clustering specifically created for Record Linkage [8].

Note that `privJedAI` supports *multi-threaded execution* via Ray[7] alongside *GPU-acceleration* for intensive vectorized computations. While single-threaded execution is the default mode, the framework heavily relies on optimized vectorized libraries to significantly enhance processing speed. Moreover, the library can be easily installed through PyPI and can be extended by the community, with comprehensive documentation and tutorials ensuring accessibility for researchers and novice users alike.

[1] https://github.com/datasciencecampus/pprl_toolkit
[2] https://linkja.github.io
[3] https://git.informatik.uni-leipzig.de/dbs/pprl/primat
[4] https://github.com/usc-isi-i2/amppere
[5] https://github.com/data61/anonlink.git
[6] https://github.com/facebookresearch/faiss
[7] https://docs.ray.io/en/latest/index.html

Figure 3: Execution of privJedAI on a Jupyter Notebook.

4 RELATED SYSTEMS

`privJedAI` unifies a comprehensive suite of existing PPRL techniques into a single, cohesive library. In contrast, existing open-source PPRL tools often suffer from fragmentation, limited scope, or lack of active maintenance. For instance, *Linkja* is a Java-based tool that requires users to compile and execute separate programs for each individual step of the pipeline, significantly hindering the seamless execution of complex workflows. Furthermore, older repositories such as the *PPRL-toolkit* are currently archived on GitHub and are no longer actively maintained. While modern frameworks like *Anonlink* and *PRIMAT* offer excellent documentation and robust capabilities, they do not integrate the same extensive variety of methods, such as the fusion of generic record linkage meta-blocking with secure PPRL encodings, within a single ecosystem. Finally, while *AMPPERE* introduces advanced matching capabilities utilizing Homomorphic Encryption, `privJedAI` currently prioritizes highly scalable ANNS methods, with plans to integrate Homomorphic Encryption in future releases.

5 DEMONSTRATION SCENARIO

This demonstration showcases `privJedAI` through live user interaction, highlighting its unique capabilities and ease of use. Attendees will be prompted to select the input datasets to be processed through the end-to-end pipeline. Notably, the framework's methods are entirely unsupervised, requiring no pre-labeled training data. Instead, the user is seamlessly guided to configure the necessary parameters and execute the workflow steps in the correct sequential order to satisfy their specific use case.

The demonstration will be conducted via a Jupyter Notebook interface (Fig. 3), which clearly outlines all mandatory and optional workflow steps alongside their respective methods. During execution, each method provides real-time status logging accompanied by progress bars. Upon the completion of each step, the notebook automatically reports detailed metrics regarding both matching effectiveness and computational efficiency.

To comprehensively evaluate performance across different hardware configurations, we will replicate the baseline notebook execution on two distinct platforms. First, we will utilize Kaggle to demonstrate the significant runtime speed-up achieved by our GPU-Accelerated methods. Second, we will execute the identical workflow on a 32-core server hosted by the AI Team of the National and Kapodistrian University of Athens, Greece, to showcase multicore CPU scalability. By running a concurrent system monitor during these parallel executions, we will illustrate the exact memory footprint and computational overhead, providing a transparent view of the costs associated with these hardware accelerations.

6 CONCLUSIONS

We presented `privJedAI`, a comprehensive open-source, end-to-end PPRL framework. Its modular architecture flexibly integrates state-of-the-art methods, while hardware acceleration successfully mitigates Python's runtime overhead. `privJedAI` offers the configuration flexibility required to address diverse user needs while remaining accessible to novices. After demonstrating these capabilities live, we plan to incorporate additional workflow methods and Homomorphic Encryption-based matching into the library.

Acknowledgments. This work was partially funded by the EU Horizon project RECITALS (Grant Agreement 101168490).

REFERENCES

[1] V. Christen, T. Häntschel, P. Christen, and E. Rahm. 2023. Privacy-preserving record linkage using autoencoders. *J. Data Sc. and Anal.* 15 (2023), 347—-357.
[2] M. Franke, Z. Sehili, and E. Rahm. 2019. PRIMAT: a toolbox for fast privacy-preserving matching. *PVLDB* 12, 12 (2019), 1826–1829.
[3] A. Gkoulalas-Divanis, D. Vatsalan, D. Karapiperis, and M. Kantarcioglu. 2021. Modern Privacy-Preserving Record Linkage Techniques: An Overview. *TIFS* 16 (2021), 4966–4987.
[4] D. Karapiperis, A. Gkoulalas-Divanis, and V. Verykios. 2016. LSHDB: a parallel and distributed engine for record linkage and similarity search. In *ICDMW*. 1–4.
[5] Y. Malkov and D. Yashunin. 2018. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *TPAMI* 42, 4 (2018), 824–836.
[6] F. Neuhof, M. Fisichella, G. Papadakis, K. Nikoletos, N. Augsten, W. Nejdl, and M. Koubarakis. 2024. Open benchmark for filtering techniques in entity resolution. *VLDB J.* 33, 5 (2024), 1671–1696.
[7] M. Norouzi, A. Punjani, and D. J. Fleet. 2014. Fast Exact Search in Hamming Space With Multi-Index Hashing . *TPAMI* 36, 06 (2014), 1107–1119.
[8] G. Papadakis, V. Efthymiou, E. Thanos, O. Hassanzadeh, and P. Christen. 2023. An analysis of one-to-one matching algorithms for entity resolution. *VLDB J.* 32, 6 (2023), 1369–1400.
[9] T. Ranbaduge, D. Vatsalan, and P. Christen. 2020. Secure Multi-party Summation Protocols: Are They Secure Enough Under Collusion? *TDP* 13, 1 (2020), 25–60.
[10] T. Ranbaduge, D. Vatsalan, and M. Ding. 2024. Privacy-Preserving Deep Learning Based Record Linkage. *TKDE* 36, 11 (2024), 6839–6850.
[11] C. Regueiro, O. Lage, E. Jacob, and J. Astorga. 2025. Challenges and future research directions in secure multi-party computation for resource-constrained devices and large-scale computations. *J. of Inf. Sec.* 24, 27 (2025).
[12] S. Stammler, T. Kussel, P. Schoppmann, F. Stampe, G. Tremper, S. Katzenbeisser, K. Hamacher, and M. Lablans. 2020. Mainzelliste SecureEpiLinker (MainSEL): Privacy-Preserving Record Linkage using Secure Multi-Party Computation. *Bioinformatics* (2020).
[13] C. Toth, E. Durham, M. Kantarcioglu, Y. Xue, and B. Malin. 2014. SOEMPI: A secure open enterprise master patient index software toolkit for private record linkage. In *AMIA*, Vol. 2014. 1105.
[14] A. Vidanage, P. Christen, T. Ranbaduge, and R. Schnell. 2023. A Vulnerability Assessment Framework for Privacy-preserving Record Linkage. *TPS* 26, 3 (2023).
[15] S. Yao, Y. Ren, D. Wang, Y. Wang, W. Yin, and L. Yuan. 2023. SNN-PPRL: A secure record matching scheme based on siamese neural network. *J. Inf. Sec. and Appl.* 76 (2023), 103529.
[16] Y. Yao, T. Ghai, S. Ravi, and P. Szekely. 2021. AMPPERE: A Universal Abstract Machine for Privacy-Preserving Entity Resolution Evaluation. In *CIKM*.
[17] W. Yin, L. Yuan, Y. Ren, W. Meng, D. Wang, and Q. Wang. 2024. Differential Cryptanalysis of Bloom Filters for Privacy-Preserving Record Linkage. *TIFS* 19 (2024), 6665–6678.