Journal of Information and Data Management, 2026, 17:1, doi: 10.5753/jidm.2026.7430
This work is licensed under a Creative Commons Attribution 4.0 International License.

Measuring Blocking Precision More Accurately in Entity Resolution

Dimas Cassimiro do N. Filho [**Universidade Federal do Agreste de Pernambuco (UFAPE)** | ***dimas.cassimiro@ufape.edu.br***]
Vitor Alan Bezerra da Silva [**Universidade Federal do Agreste de Pernambuco (UFAPE)** | ***vitoralan.bs@gmail.com***]

Federal University of Agreste de Pernambuco (UFAPE), Av. Bom Pastor, s/n, Garanhuns, PE, Brazil.

Received: 20 January 2026 • **Published:** 27 November 2026

Abstract Entity resolution plays a fundamental role in data integration, with the objective of detecting records that correspond to the same physical entity. To enhance computational efficiency, blocking methods are commonly employed to limit the number of pairwise comparisons. Nevertheless, established evaluation measures like Pair Quality (PQ) do not consider duplicate comparisons, which may lead to misleading evaluations of a blocking strategy's performance. In this work, we propose the PQ^* metric, which offers a more faithful precision estimate by factoring out the influence of redundant comparisons. We also present the PQ^*C and $\widehat{PQ^*}E$ algorithms for the efficient exact computation and estimation of PQ^*, respectively. Our experiments demonstrate that, across all tested datasets, PQ and PQ^* produce distinct outcomes in every scenario involving multiple blocking keys. Moreover, the discrepancy between the two metrics grows as the number of indexing keys increases.

Keywords:
Entity Resolution, Blocking, Pairs Quality, Indexing

1 Introduction

Entity Resolution (ER) is a fundamental task in data integration, responsible for identifying records that represent the same real-world entity Christen and Christen [2012]. This task is particularly challenging due to the presence of dirty and noisy data Araújo *et al*. [2019], variations in writing formats, and the absence of unique identifiers. In applications such as healthcare systems, governmental databases, and population census consolidation, efficiency and precision in ER are essential to ensure data integrity Getoor and Machanavajjhala [2012].

Due to the high computational cost of ER, blocking techniques are widely used to reduce the number of pairwise entity comparisons Li *et al*. [2020]; Papadakis *et al*. [2020]. The goal of blocking is to group similar records into smaller blocks, within which comparisons are performed, avoiding the need for an exhaustive pairwise comparison process among all possible pairs Mestre *et al*. [2017a]; Araújo *et al*. [2019]. We illustrate a simplified ER workflow in Figure 1.

The evaluation of blocking precision is often performed using the metrics PC (Pairs Completeness), RR (Reduction Ratio), and PQ (Pairs Quality) Nascimento *et al*. [2020a], with the latter being responsible for measuring the proportion of correctly identified duplicate pairs relative to the total number of record pairs to be compared Papadakis *et al*. [2016b, 2020]. However, the PQ metric has limitations, as it does not account for redundant comparisons, which can lead to inaccurate conclusions about the precision of the blocking scheme. To mitigate this limitation, we propose the Accurate Precision metric, denoted by PQ^*, which improves the calculation of blocking precision by disregarding redundant comparisons. Unlike PQ, the proposed accurate metric (PQ^*) provides a more realistic view of the quality of the blocking result, considering that, in a real ER scenario, redundant comparisons are not actually computed, avoiding unnecessary processing in the comparison stage Christen and Christen [2012].

In this article, we aim to propose different approaches for calculating the accurate precision (PQ^*) of the blocking result produced in ER tasks. More specifically, we intend to investigate three main questions:

Question 1: How to optimize the calculation of PQ^*?

Question 2: How to estimate the value of PQ^* in the context of large volumes of data?

Question 3: What is the difference in complexity between approaches used to calculate and estimate PQ^*?

Considering the aforementioned questions, we proposed the PQ^*C algorithm, optimized to calculate PQ^* efficiently; and the $\widehat{PQ^*}E$ algorithm, designed to estimate PQ^* in a scalable way. The obtained experimental results demonstrate that, in practice, the PQ and PQ^* metrics produced different results across all evaluated datasets. Furthermore, the difference between the results of the two metrics increases as more blocking keys are used in the indexing process and, consequently, produce more redundant comparisons. This article is an extended version of a previously published conference paper Nascimento and Silva [2025]. In this version, we provide further details regarding the proposed metric, including illustrative examples, practical guidelines and takeaways, and a more detailed discussion of related works.

In summary, the following contributions are presented in this article:

- Proposal of a new metric for calculating the accurate precision of the blocking result in the ER task;

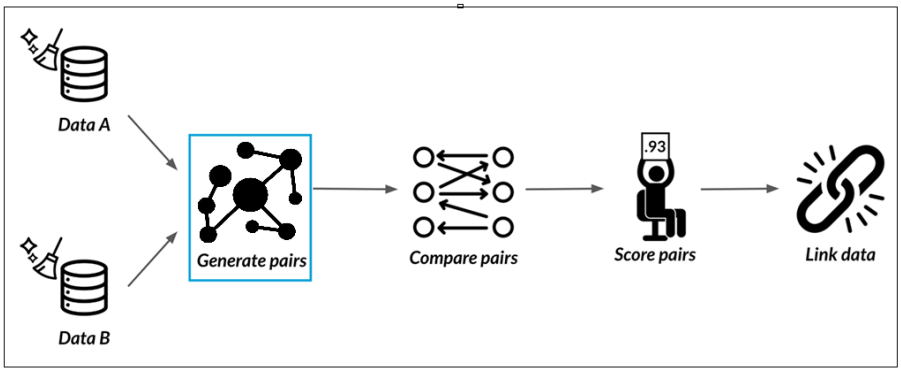

Figure 1. Simplified ER workflow.

- Proposal of algorithms for the exact calculation and estimation of the proposed metric, along with the theoretical asymptotic analysis of their complexities;
- Experimental evaluation of the results of the proposed metrics considering datasets with different characteristics;
- A discussion of practical guidelines and takeaways regarding the usage of the proposed metric.

Practical Implications of the Proposed Metric. This work establishes a formal link between the proposed PQ^* metric and the actual computational efficiency gained during the comparison step of ER. By analytically discounting redundant comparisons, PQ^* provides a direct theoretical measure of the true positive rate among *unique* candidate pairs. This allows for a more rigorous assessment of a blocking scheme's ability to prune the search space effectively. We demonstrate that PQ^* is a lower bound for PQ under any blocking strategy that can generate redundancy.

Implications for Blocking Strategy Design. The adoption of PQ^* has immediate practical implications for the design and tuning of blocking strategies. By focusing on the accurate precision of unique comparisons, data engineers can make more informed decisions when selecting blocking functions, determining the number of blocking keys, and setting similarity thresholds. Our findings suggest that strategies optimized solely for high traditional PQ might be overfitting to redundant, easy-to-match pairs. In contrast, PQ^*-aware optimization may promote the generation of a diverse and non-redundant candidate set, which is crucial for achieving high recall while maintaining efficiency in large-scale, real-world ER pipelines. Future blocking algorithms can integrate PQ^* estimation directly into their learning or optimization loops to produce higher-quality blocks.

The remainder of this article is structured as follows: Section 2 presents the theoretical foundation of the article. Section 3 presents the formalization of the proposed metric. In Section 4, we propose algorithms to calculate PQ, as well as to calculate and estimate the value of the proposed metric (PQ^*). In Section 5, we detail the experiments performed and analyze the results obtained. In Section 6, we present the main related works. Finally, in Section 7, we discuss the conclusions of the study and suggest directions for future work.

2 Theoretical Background

This section introduces the main concepts and metrics underlying the evaluation of blocking effectiveness and efficiency in ER.

2.1 Pair Completeness and Quality

The effectiveness of an ER pipeline is typically measured along two key dimensions: *effectiveness*, which relates to the quality and completeness of the matches found, and *efficiency*, which concerns the computational resources required to process the data. Achieving a balance between these two dimensions is a persistent challenge, as more sophisticated matching algorithms that improve accuracy often come at the cost of higher computational overhead. Consequently, research in ER frequently focuses on developing scalable techniques that maintain high-quality results while operating within practical time and resource constraints.

To evaluate the efficiency and effectiveness of blocking approaches, metrics such as *Pair Completeness* (PC) and *Pairs Quality* (PQ) are used. The PC metric measures the proportion of matching entity pairs that were included in the blocks relative to the total number of true pairs existing. Formally, PC is defined as:

$$PC = \frac{|DP|}{|TP|} \tag{1}$$

where $|DP|$ represents the number of detected duplicate pairs and $|TP|$ the total number of duplicate pairs existing in the dataset(s).

In turn, the PQ metric measures the quality of the comparisons performed, i.e., the fraction of comparisons that actually correspond to duplicate pairs. Its formal definition is given by:

$$PQ = \frac{|DP|}{|CP|} \tag{2}$$

where $|CP|$ represents the total number of comparisons performed. A high PC value indicates that the approach managed to recover most of the duplicate entities, while a high PQ value suggests that most of the comparisons performed were relevant.

2.2 Redundant Comparisons and Superfluous Comparisons

In the context of ER, comparisons performed during the blocking process can be classified into different categories. Redundant comparisons occur when the same pair of records is evaluated multiple times, increasing the computational cost without improving the resolution quality Papadakis *et al*. [2016a]. This phenomenon is particularly common in blocking schemes that combine multiple keys via disjunctions, where the same record pair can be allocated to different blocks due to different grouping criteria. Furthermore, it occurs in blocking techniques that generate overlapping blocks, such as those based on sliding windows, which cause records to be included in multiple blocks, resulting in repetitive and unnecessary comparisons.

On the other hand, *superfluous* comparisons correspond to those involving pairs of records that do not represent the same entity in the real world. The goal of efficient blocking approaches, such as *meta-blocking* Papadakis *et al*. [2013], is to reduce both redundant and superfluous comparisons, with the intention that the blocks contain more pairs relevant to the ER process.

3 PQ^* Metric: Accurate Blocking Precision

In this section, a new metric for calculating the precision of blocking results in ER tasks is proposed.

Although PQ provides a useful estimate of comparison precision, this metric is sensitive to redundant comparisons. If the same pair (e, e') is repeatedly compared in different blocks, this fact will affect the assessment of blocking quality, as PQ treats all instances of record comparisons as independent, even if they are related to the same record pair.

The PQ^* metric (Accurate Blocking Precision) is a variation of the PQ (Pairs Quality) metric that seeks to measure the precision of comparisons generated by the blocking strategy by eliminating the impact of redundant comparisons. While PQ is defined as the proportion of duplicate pairs relative to the total pairs compared, PQ^* improves this measure by excluding artificial redundancies introduced by data partitioning.

This improvement is particularly relevant for evaluating modern, multi-key or overlapping blocking strategies, such as canopy clustering or token-based blocking, which inherently generate redundant candidate pairs. The traditional PQ metric can significantly overestimate precision in these scenarios, whereas PQ^* provides a more realistic assessment that correlates directly with the computational efficiency gained in the subsequent, more expensive matching phase. By focusing on unique comparisons, PQ^* offers a clearer view of how well the blocking step filters the search space.

3.1 Formal Definition

Let $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$ be a set of blocks generated by a blocking method, where each block B_i contains a subset of records. The total number of comparisons performed within these blocks is given by:

$$C = \sum_{B \in \mathcal{B}} |B|(|B| - 1)/2$$

However, due to possible overlap between blocks, the same pair of records may be compared multiple times. Let $\mathcal{U}$ be the set of unique pairs effectively compared by the blocking scheme, i.e., without considering duplicate comparisons. The total number of unique comparisons can be expressed as $|\mathcal{U}|$, and the number of correctly identified duplicate pairs within these comparisons is denoted by $|DupSet|$. Thus, the PQ^* metric is defined as:

$$PQ^* = \frac{|DupSet|}{|\mathcal{U}|}$$

such that $|DupSet|$ represents the number of correctly identified duplicate pairs and $|\mathcal{U}|$ corresponds to the number of unique pairs compared, removing multiple instances of the same pair in different blocks.

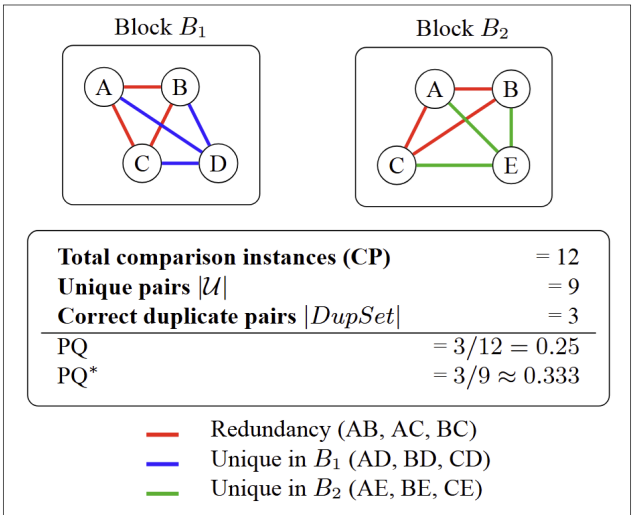

Figure 2. Illustrating the proposed metric applied to blocking results.

Example 1. *Figure 2 illustrates a simplified blocking scenario in an ER task, designed to highlight the impact of redundant comparisons on the traditional Pairs Quality (PQ) metric and to motivate the proposed PQ^* metric. The example consists of two blocks, denoted as B_1 and B_2. Each block groups a subset of records that are considered similar according to a blocking criterion. Block B_1 contains the records $\{A, B, C, D\}$, while block B_2 contains the records $\{A, B, C, E\}$. Due to overlapping blocking keys, records A, B, and C appear in both blocks.*

Within each block, all possible pairwise comparisons are generated. As a result, block B_1 produces the comparisons

$$\{AB,\ AC,\ BC,\ AD,\ BD,\ CD,\}$$

and block B_2 produces

$$\{AB,\ AC,\ BC,\ AE,\ BE,\ CE.\}$$

Altogether, the blocking strategy generates $|CP| = 12$ comparison instances. The edges in the figure are color-coded to

emphasize different types of comparisons. Red edges represent redundant comparisons*, namely the pairs AB, AC, and BC, which are generated in both B_1 and B_2. These comparisons do not contribute new information but increase the computational cost. Blue edges correspond to comparisons that are unique to block B_1, while green edges represent comparisons unique to block B_2.*

Assume that the true duplicate pairs in the dataset are exactly $\{AB, AC, BC\}$. Hence, the number of correctly identified duplicate pairs is $|DupSet| = 3$. Using the traditional PQ definition, which counts all comparison instances (including redundancies), we obtain:

$$PQ = \frac{|DupSet|}{|CP|} = \frac{3}{12} = 0.25.$$

However, this value underestimates the effective quality of the blocking, since the duplicate pairs are repeatedly counted due to block overlap. If we consider only unique *record pairs, the total number of distinct comparisons is*

$$|\mathcal{U}| = 9,$$

namely $\{AB, AC, BC, AD, BD, CD, AE, BE, CE\}$. When redundancy is removed, the accurate blocking precision becomes:

$$PQ^* = \frac{|DupSet|}{|\mathcal{U}|} = \frac{3}{9} \approx 0.333.$$

This example clearly demonstrates that redundant comparisons artificially inflate the denominator of the PQ metric, leading to a distorted assessment of blocking quality. By focusing on unique comparisons only, PQ^* provides a more faithful measure of the true precision of a blocking strategy, especially in scenarios involving overlapping and/or multi-key blocking.

3.1.1 Final Remarks

Both redundancy growth models reinforce the theoretical motivation for PQ^*. As the number of blocking keys increases, the traditional PQ metric becomes increasingly dominated by redundant comparison instances, leading to an overly pessimistic assessment of blocking effectiveness. In contrast, PQ^* remains sensitive only to the expansion of the unique comparison space, providing a more accurate, interpretable, and operationally meaningful evaluation of blocking quality in modern, large-scale ER pipelines.

4 Approaches for Calculating PQ and PQ^*

This section presents approaches for calculating PQ and PQ^*, as well as a heuristic approach to estimate the calculation of PQ^* in order to process large volumes of data in a scalable manner.

To facilitate the proposal of the algorithms, we use the notion of k-combinations of a set, as presented in Definition 1.

Definition 1 (k-combinations of a Set)**.** *Let S be a set. We denote by $\binom{S}{k}$ the set of all k-combinations of the set S, i.e., $\binom{S}{k} = \{s \in 2^S \mid |s| = k\}$, where 2^S is the power set of S and $|S|$ represents the cardinality of the set S.*

4.1 PQC Algorithm

Algorithm 1 (*PQC - PQ Calculator*) calculates the *PQ* (pairs quality) metric, which measures the proportion of duplicate pairs in a set of blocks $\mathcal{B}$. The algorithm receives as input a set of blocks $\mathcal{B}$ and returns a numerical value representing the quality of the pairs. Initially, the structures `DupSet`, an empty set that will store the found duplicate pairs, and `Cont`, a counter variable initialized to zero, are created (lines 2 and 3).

From line 3, the algorithm iterates through each block $B \in \mathcal{B}$. For each block, the variable `Cont` is updated in line 4 to store the total number of pairs that can be formed within the block. This calculation is given by the combinatorial formula, which represents the number of record pairs (e, e') within block B. Then, the algorithm iterates through all possible pairs within B (line 5), using the notation $\binom{B}{2}$, which represents all combinations of two elements from the block. For each pair (e, e'), it is checked in line 6 if the elements are equivalent ($e_1 \equiv e_2$). If this condition is true, the pair is added to the `DupSet` set (line 7).

After iterating over all blocks, the algorithm returns, in line 8, the *PQ* metric. This equation represents the proportion of duplicate pairs found relative to the total pairs produced by the blocking.

Algorithm 1: PQC - PQ Calculator

```
   Input: B set of blocks
   Output: PQ pairs quality
1  DupSet ← ∅
2  Cont ← 0
3  foreach B ∈ B do
4  |  Cont ⟵ 2^-1 * |B| * (|B| - 1))
5  |  foreach (e, e') ∈ (B 2) do
6  |  |  if e ≡ e' then
7  |  |  |  DupSet.add(ordered(e, e'))
8  return |DupSet|/Cont
```

4.2 PQ^*C Algorithm

Algorithm 2 (PQ^***C**) follows a principle similar to **PQC**, but it seeks to eliminate redundant comparisons. To achieve this, in addition to the `DupSet`, the structure `comparisonsSet` is introduced, a set used to store already compared pairs, ensuring that each pair is evaluated only once. The algorithm iterates through each block B in the set $\mathcal{B}$ and, for each pair (e, e') generated by the combinations within the block, the ordered representation of the pair is added to `comparisonsSet`. This storage prevents the same pair from being evaluated repeatedly in different blocks, reducing comparison redundancy. If the elements of the pair are considered equivalent, it is added to the `DupSet`. At the end, the

PQ metric is calculated as the ratio between the number of duplicate pairs stored in `DupSet` and the total number of record pairs in `comparisonsSet`. In this way, the PQ^*C algorithm produces a more accurate metric regarding the precision of the blocking result.

Algorithm 2: PQ^*C - PQ^* Calculator

Input: $\mathcal{B}$ set of blocks
Output: PQ pairs quality

```
1 DupSet ← ∅
2 comparisonsSet ← ∅
3 foreach B ∈ B do
4     foreach (e, e') ∈ (B 2) do
5         comparisonsSet.add(ordered(e, e'))
6         if e ≡ e' then
7             DupSet.add(ordered(e, e'))
8 return |DupSet|/|comparisonsSet|
```

Asymptotic analysis. The algorithm iterates through all blocks and, for each block, generates all combinations of two elements. Let n be the number of blocks and m the average block size. The outer loop iterates through n blocks, resulting in $O(n)$. The inner loop generates all combinations of two elements within a block of average size m, which results in complexity $O(m^2)$, considering that inserting the record pairs into the $ComparisonsSet$ can be performed in $O(1)$, using a hash-based data structure. Thus, the total complexity is approximately $O(nm^2)$.

4.3 Comparison between PQC and PQ^*C

Although the PQC algorithm has the same asymptotic complexity as PQ^*C, it avoids storing comparison pairs explicitly, resulting in substantially lower memory consumption in practice. In the PQ^*C algorithm, the `comparisonsSet` is used to store compared pairs, ensuring that each comparison is performed only once. However, this approach consumes memory proportional to the number of unique pairs, which can be prohibitive for large volumes of data. PQC, on the other hand, eliminates the need to store these comparisons, making it more efficient in terms of memory usage.

However, PQC has the disadvantage of not disregarding redundant comparisons when calculating the *PQ* (*pairs quality*) metric. As a result, the calculated value can be distorted by the inclusion of repeated pairs, which may compromise the accuracy level of the metric, depending on the data distribution.

4.4 $\widehat{PQ^*}E$ Algorithm

This section proposes an algorithm to estimate the value of the PQ^* metric based on a sampling technique. The objective is to develop an algorithm called $\widehat{PQ^*}E$ that is much more efficient than the strategy adopted by Algorithm 2 and that produces a good approximation of the PQ^* metric. Formally, we aim for $T_{exec}(\widehat{PQ^*}E(\mathcal{B})) \ll T_{exec}(PQ^*C(\mathcal{B}))$ and $\widehat{PQ^*}E(\mathcal{B})) \simeq PQ^*C(\mathcal{B})$, where $\mathcal{B}$ represents a set of blocks received as input.

Algorithm 3 ($\widehat{PQ^*}E$) estimates the value of the PQ^*E metric, considering a set of blocks $\mathcal{B}$ and using a random sampling of proportion α. The algorithm receives as input a set of blocks $\mathcal{B}$ and a value α that defines the fraction of blocks to be considered in the estimate. The output of the algorithm is the estimated value of $\widehat{PQ}^*$. Initially, the structures `DupSet`, a set to store identified duplicate pairs, and `comparisonsSet`, a set to record the performed comparisons, are created (lines 1-2).

The sampling occurs in line 3, where a subset $\mathcal{B}^-$ containing $\alpha \cdot |\mathcal{B}|$ blocks from $\mathcal{B}$ is selected. Then, the algorithm iterates through each block $B \in \mathcal{B}^-$ (line 4). For each block, all possible pairs (e, e') within block B are generated, using the combination $\binom{B}{2}$ (line 5). Each pair is then added to the `comparisonsSet`. For each pair of records in the block, if the records are duplicates, i.e., $e \equiv e'$, then this pair is added to the `DupSet` set (lines 7-8). This check allows identifying the pairs that represent duplicates within the selected sample. After finishing the iteration over the blocks, the estimator $\widehat{PQ}^*$ is calculated and returned in line 9.

Algorithm 3: $\widehat{PQ^*}$E - PQ^* Estimator

Input: $\mathcal{B}$ set of blocks, $\alpha \in [0, 1]$ sampling percentage of blocks
Output: $\widehat{PQ^*}$ (estimated accurate pairs quality)

```
1 DupSet ← ∅
2 comparisonsSet ← ∅
3 B⁻ ⟵ select (|B| * α) blocks from B
4 foreach B ∈ B⁻ do
5     foreach (e, e') ∈ (B 2) do
6         comparisonsSet.add(ordered(e, e'))
7         if e ≡ e' then
8             DupSet.add(ordered(e, e'))
9 return (|DupSet|)/|comparisonsSet|
```

Asymptotic Analysis. Let n be the total number of blocks and m the average block size. The cost of selecting the sample of size αn in line 6 is $O(\alpha n)$. For each selected block in the sample (line 4), $O(m^2)$ pairs are generated (line 5), resulting in a total cost of $O(\alpha n m^2)$. The insertion of record pairs into the sets occurs in $O(1)$ complexity, if we use an efficient data structure, such as a hash map or hash set. The final calculation of the estimator in line 9 is $O(1)$. Thus, the overall complexity of the algorithm is $O(\alpha n m^2)$, dominated by the generation of pairs within the blocks.

5 Discussion of the Investigated Questions

The approaches presented in this section were designed to directly address the three questions formulated in Section 4. Below, we discuss how each of the questions defined in Section 4 is answered:

- **Question 1: How to optimize the calculation of PQ^*?** This question is addressed by the proposition of the

PQ^*C algorithm, which seeks to eliminate redundant comparisons through the use of an auxiliary memory structure (`comparisonsSet`). Although this approach increases memory consumption, it significantly improves the accuracy of the calculation by avoiding repeated pair counts, producing a more accurate metric than the PQC algorithm.

- **Question 2: How to estimate the value of PQ^* in the context of large volumes of data?**
  To answer this question, we proposed the $\widehat{PQ^*E}$ algorithm, which performs an efficient estimation of the PQ^* metric using a sampling technique. By processing only a fraction of the available blocks, the algorithm reduces execution time and memory usage, offering a viable alternative for scenarios requiring the processing of large databases.
- **Question 3: What is the difference in complexity between approaches used to calculate and estimate PQ^*?**
  The asymptotic analysis of the approaches shows that the PQC algorithm has complexity $O(nm^2)$, being more memory efficient. The PQ^*C algorithm, despite maintaining similar complexity, has higher memory usage due to the explicit storage of performed comparisons. Finally, the estimator $\widehat{PQ^*E}$ presents reduced complexity $O(\alpha nm^2)$, making it suitable for fast estimates with a reasonable level of accuracy.

In summary, the research questions guided the design of three complementary algorithmic strategies, each balancing the trade-offs between computational resources (time and memory) and the fidelity of the result. The progression from the baseline PQC to the accurate PQ^*C and finally to the efficient $\widehat{PQ^*E}$ estimator provides a complete toolkit for evaluating blocking precision. Practitioners can select the most appropriate algorithm based on their specific constraints: PQC for quick, memory-efficient analysis with traditional precision; PQ^*C for exact, high-fidelity evaluation when redundant comparisons must be accounted for; and $\widehat{PQ^*E}$ for scalable analysis of large datasets where approximate results are acceptable. This section validates the initial problem decomposition and demonstrates that all posed theoretical questions have been addressed through concrete algorithmic contributions.

6 Experimental Evaluation

In this section, experiments are conducted to empirically evaluate: i) the results of the proposed metric (PQ^*); and ii) how the results of (PQ^*) differ from the classical metric (PQ) for evaluating the precision results produced by blocking techniques in the context of ER. The experiments were conducted to answer the following questions:

- **Q1:** Are there differences in the results produced by the metrics (PQ^*) and (PQ)?
- **Q2:** Do the results produced by the metrics (PQ^*) and (PQ) vary according to the number of blocking keys used?

6.1 Experimental Environment

All experiments were conducted on a computer with an Intel(R) Core(TM) Ultra 7 155H 3.80 GHz processor, 328GB of RAM, and the Windows 11 operating system. The algorithms were implemented in Java, using the commons-codec library for Soundex-based indexing.

6.2 Datasets

We selected synthetic and real-world datasets to evaluate the proposed approach, considering different characteristics and sizes. The following datasets were used in the experimental evaluation:

- **North Carolina Voters**[1]: a real dataset of voters from the state of North Carolina;
- **DS3**: a dataset containing scientific publication records Mestre *et al*. [2017b];
- **Company Names**: a dataset with company names (used to evaluate collective classification approaches for ER in Hassanzadeh *et al*. [2009]);
- **Cora**: a dataset of publication records;
- **DBLPM4**: a dataset of publication titles (also used in Hassanzadeh *et al*. [2009]);

These datasets were chosen because they present different sizes, duplicate proportions, and number of attributes, allowing for a comprehensive evaluation of the proposed algorithms.

6.3 Experiment Design

For each dataset, we applied the standard blocking technique Christen and Christen [2012], including blocking keys based on phonetic keys (Soundex) and blocking by prefixes or suffixes. For each database, the results produced by the PQ and PQ^* metrics were evaluated, considering 1, 2, and 3 blocking keys combined in a disjunctive manner.

For the calculation of PQ, the PQC algorithm (Algorithm 1) was used. For the calculation of PQ^*, we used the PQ^*C algorithm (Algorithm 2).

In the scenario where only one blocking key is used to index the database, no redundant comparisons are produced, given that standard blocking produces disjoint blocks. Conversely, when using a number $n > 1$ of blocking keys, redundant comparisons can be produced considering the blocks generated by the different blocking keys, and consequently, it is possible to obtain different results for the metrics PQ^* and PQ. However, we decided to evaluate the metric results considering only one blocking key to ensure that the implementation was correct. In this sense, the following conditions must be verified in the obtained results:

$$\big(n = 1\big) \implies \big(PQ^* = PQ\big) \tag{3}$$

$$\Diamond(n > 1 \ \wedge PQ^* > PQ) \tag{4}$$

, where $\Diamond$ denotes the possibility symbol in modal logic.

[1] https://dbs.uni-leipzig.de/research/projects/benchmark-datasets-for-entity-resolution

Dataset	Number of Blocking Keys	PQ	PQ^*
Cora	1	0.45	0.45
	2	0.20	0.35
	3	0.13	0.25
DBLPM4	1	0.22	0.22
	2	0.14	0.22
	3	0.062	0.095
Company Names	1	0.13	0.13
	2	0.09	0.13
	3	0.07	0.13
DS3	1	0.12	0.12
	2	0.03	0.04
	3	0.028	0.033
NCV	1	0.0016	0.0016
	2	0.00083	0.0016
	3	0.00079	0.0014

Table 1. Results of PQ and PQ^* considering one, two, and three blocking keys combined in a disjunctive manner.

6.4 Experimental Results

The experimental results presented in Table 1 allow us to address the research questions Q1 and Q2 formulated at the beginning of this section. Regarding **Q1**, it is observed that, for all evaluated datasets, the PQ and PQ^* metrics produce identical results when only one blocking key is used. This behavior was expected, as in this scenario the generated blocks are disjoint and, therefore, there are no redundant comparisons between record pairs, which guarantees that both metrics compute the same set of comparisons in the denominator of the respective metrics.

When considering multiple blocking keys combined by disjunction (i.e., $n = 2$ or $n = 3$), the metrics produce different results. In all cases, the values of PQ^* exceed those of PQ. This difference is explained by the fact that the classical calculation of PQ considers redundant comparisons, while PQ^* is designed to eliminate such redundancies from the total comparison count. Thus, the PQ^* metric more accurately reflects the impact of blocking on the quality of ER, especially in scenarios where there is overlap between blocks or multiple blocking keys are used to index the records.

Regarding **Q2**, the results confirm that the use of multiple blocking keys tends to accentuate the difference between the metrics. As the number of keys increases, a reduction in PQ values is observed, while the PQ^* values remain stable or decrease with less intensity. This reinforces the ability of PQ^* to mitigate the impact of redundancy, capturing more fairly the contribution of additional blocks to the identification of correct pairs.

6.5 Takeaways and Practical Guidelines

The observed divergence between the metrics has direct practical implications. The results highlight that the gap between PQ and PQ^* is dataset-dependent and closely related to the structural properties of the blocks generated by each blocking strategy. Datasets such as *Cora* and *Company Names* exhibit a pronounced divergence between the two metrics as the number of blocking keys increases, indicating a higher degree of overlap among blocks and, consequently, a larger volume of redundant comparisons. In contrast, datasets with sparser duplication patterns or more heterogeneous records, such as *DS3* and *NCV*, present smaller absolute differences, although the same qualitative trend is preserved. This observation suggests that PQ^* is particularly informative in scenarios where blocking strategies aggressively trade redundancy for recall, as it isolates the true contribution of unique comparisons.

Therefore, the choice of metric fundamentally alters the interpretation of a blocking strategy's performance. A system optimized for a high traditional PQ might prioritize generating large, non-overlapping blocks, potentially at the cost of recall. A system optimized for PQ^*, however, would be less penalized for employing multiple, overlapping keys, which is often necessary to achieve high recall in heterogeneous data. Our results demonstrate that PQ^* provides a more balanced and operationally relevant measure, as it aligns with the actual computational workload and utility of the candidate pair set generated for the subsequent, expensive matching step.

Finally, these findings emphasize the importance of using redundancy-aware evaluation metrics when comparing blocking techniques. Relying solely on the classical PQ metric may lead to overly misleading conclusions regarding the quality of blocking schemes that employ multiple or overlapping keys, as the metric penalizes repeated comparisons rather than ineffective filtering. By contrast, PQ^* provides a more realistic estimate of blocking precision by focusing on the set of unique candidate pairs that effectively reach the matching stage. As a result, PQ^* constitutes a more appropriate metric for guiding the design and tuning of modern blocking approaches, particularly in large-scale ER pipelines where computational efficiency is a critical concern.

7 Related Work

The task of ER has been extensively investigated in research conducted over the past decades Nascimento *et al*. [2016]; Papadakis *et al*. [2020]; Li *et al*. [2020], given its importance in contexts such as database integration, population census application, and fraud detection.

One of the most commonly used approaches for evaluating the quality of linking algorithms is based on traditional classification measures Nascimento *et al*. [2020b], such as precision, recall, and F-measure. However, these metrics may not be suitable in contexts with a significant imbalance between matching and non-matching pairs, as discussed in Hand and Christen [2018], where the authors argue that using the F-measure may not be appropriate when applied to ER problems. This occurs mainly due to the large asymmetry between the number of true positive pairs (matches) and true negative pairs (non-matches), which can lead to an overestimation of classifier quality. To solve this problem, methods need to classify the same number of record pairs; otherwise, the evaluation of the classification ceases to be fair.

Several previous works have explored the challenges associated with ER and its evaluation metrics. Classic references in the field of ER Elmagarmid *et al*. [2007]; Christen

and Christen [2012] discuss the main algorithms, metrics, and challenges in the area. In turn, more robust blocking techniques, based on the pruning of blocking graphs, have been proposed in Gagliardelli *et al*. [2022]; Papadakis *et al*. [2014, 2013], where the blocking metrics PQ (Pairs Quality), PC (Pairs Completeness), Reduction Ratio (RR), and the aggregated cardinality of the blocks (i.e., the number of comparisons produced by all blocks generated in the indexing stage) are employed. Other works Zeakis *et al*. [2023]; Li *et al*. [2024] have focused on evaluating pre-trained models, e.g., BERT-based models, in the context of ER. For this purpose, the evaluation metrics *Blocking Recall* (equivalent to PC) and *scalability* are explored.

More recent articles Nanayakkara *et al*. [2025]; Christen *et al*. [2025]; Christen and Goiser [2007] substantially advance the evaluation of ER systems by proposing, refining, and critically analyzing metrics that capture both effectiveness and efficiency aspects of the ER process. Nanayakkara et al. Nanayakkara *et al*. [2025] address the problem of evaluating ER in fully unsupervised settings, where ground truth labels are unavailable or prohibitively expensive to obtain. The main contribution of their work lies in the proposal of evaluation strategies that rely on intrinsic properties of the ER process rather than external annotations. Instead of directly measuring traditional effectiveness metrics such as precision and recall, the authors focus on consistency, stability, and agreement patterns induced by similarity functions and clustering behavior. By analyzing how ER outcomes vary under perturbations of similarity thresholds and comparison structures, their work introduces an alternative perspective on ER evaluation that is particularly relevant for large-scale, real-world deployments.

Christen Christen *et al*. [2025] introduces the *Consistent Record Linkage* (CRL) measure, a novel evaluation metric designed to address a critical flaw in traditional assessment methods for supervised record linkage classifiers. Standard metrics like Precision, Recall, and the F-measure are calculated from a single confusion matrix based on a specific classification threshold. This can lead to inconsistent evaluations, as different classifiers can yield identical metric values from vastly different confusion matrices, obscuring important trade-offs between false positives and false negatives. Their CRL measure overcomes this by adopting an application-centric approach by providing two crucial, integrated insights in a single score: 1) the **average linkage quality** (F^*) within acceptable operational bounds, and 2) the **robustness** of the classifier, quantified by the size of the threshold interval where it performs reliably. A larger interval indicates greater insensitivity to threshold calibration or data drift, a vital property for real-world deployment. Consequently, the CRL measure enables a fair, transparent, and practically relevant comparison of classifiers.

Lastly, Foxcroft *et al*. [2024] investigates the critical yet under-reported impact of the training data class ratio (matching to non-matching pairs) on the performance of record linkage models. Through extensive experiments with diverse datasets and model architectures, it demonstrates that varying this ratio in training and testing data significantly and predictably alters standard performance metrics (precision, recall, and F_1), with higher class imbalance leading to lower reported performance. The study concludes by offering practical recommendations, emphasizing the necessity of documenting the class ratio to ensure reproducibility and advocating for overestimating the imbalance when the deployment environment's ratio is unknown.

While the contributions presented in Christen *et al*. [2025]; Foxcroft *et al*. [2024] primarily focus on refining evaluation metrics for the *matching/classification* phase or on devising unsupervised evaluation strategies for the overall ER result (Nanayakkara *et al*. [2025]), this article shifts the focus upstream to a critical yet often overlooked component: the *blocking* stage. Specifically, it addresses a fundamental flaw in the traditional Pairs Quality (PQ) metric used to evaluate blocking effectiveness, which is distorted by redundant comparisons. The proposed PQ^* metric and its associated algorithms provide a more accurate and operationally faithful measure of blocking precision by discounting these redundancies, thus offering a direct and improved assessment of a blocking scheme's ability to efficiently prune the candidate pair search space.

8 Conclusions and Future Works

The central contribution of this work is the formalization and empirical validation of PQ^* as a more accurate and operationally faithful metric for assessing blocking effectiveness than the traditional PQ. By explicitly identifying and discounting redundant comparisons, PQ^* offers a theoretically grounded and more interpretable evaluation of blocking efficiency, closely reflecting the actual computational workload propagated to the subsequent and typically more expensive matching phase. This contribution is particularly significant for the design, evaluation, and fine-tuning of modern multi-key blocking strategies in large-scale ER pipelines, where redundancy and comparison cost play a critical role.

We also proposed approaches to calculate and estimate the PQ^* metric, which evaluates the effective precision produced by blocks generated in ER tasks. The conducted experiments demonstrated that, for all evaluated datasets, the PQ and PQ^* metrics yield identical values when a single blocking key is employed, while producing distinct results in all scenarios where more than one blocking key is used. Furthermore, the experimental results show that combining multiple blocking keys in a disjunctive manner tends to increase the gap between the values of PQ and PQ^*, reflecting the growing impact of redundant comparisons.

Based on these findings, we outline several directions for future work. First, an interesting research avenue consists of adapting the proposed algorithms to *streaming* scenarios, in which data arrives continuously and cannot be entirely stored in memory. Additionally, we plan to extend the experimental evaluation by conducting a more comprehensive analysis of the influence of the parameter α on the estimated value of PQ^*.

We also intend to explore the relationship between PQ^* and matching quality in subsequent stages of the ER pipeline, analyzing how reductions in redundant comparisons impact not only efficiency but also precision, recall, and fairness. Finally, future work includes the investigation of learning-

based and adaptive blocking strategies guided by PQ^*. In this context, the proposed metric could be directly incorporated into optimization objectives Nascimento *et al*. [2025], enabling the automatic selection or weighting of blocking keys according to their effective contribution to reducing computational cost while preserving candidate quality.

References

Araújo, T. B., Pires, C. E. S., Mestre, D. G., Nóbrega, T. P. d., Nascimento, D. C. d., and Stefanidis, K. (2019). A noise tolerant and schema-agnostic blocking technique for entity resolution. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 422–430.

Christen, P. and Christen, P. (2012). *The data matching process*. Springer.

Christen, P. and Goiser, K. (2007). Quality and complexity measures for data linkage and deduplication. In *Quality measures in data mining*, pages 127–151. Springer.

Christen, P., Ziyad, S., and Nanayakkara, C. (2025). Consistently evaluating record linkage classifiers. *ACM Journal of Data and Information Quality*.

Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007). Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16.

Foxcroft, J., Christen, P., and Antonie, L. (2024). Class ratio and its implications for reproducibility and performance in record linkage. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 194–205. Springer.

Gagliardelli, L., Papadakis, G., Simonini, G., Bergamaschi, S., Palpanas, T., *et al*. (2022). Generalized supervised meta-blocking. *Proceedings of the VLDB Endowment*, 15(9):1902–1910.

Getoor, L. and Machanavajjhala, A. (2012). Entity resolution: theory, practice & open challenges. *Proceedings of the VLDB Endowment*, 5(12):2018–2019.

Hand, D. and Christen, P. (2018). A note on using the f-measure for evaluating record linkage algorithms. *Statistics and Computing*, 28:539–547.

Hassanzadeh, O., Chiang, F., Lee, H. C., and Miller, R. J. (2009). Framework for evaluating clustering algorithms in duplicate detection. *Proceedings of the VLDB Endowment*, 2(1):1282–1293.

Li, B.-H., Liu, Y., Zhang, A.-M., Wang, W.-H., and Wan, S. (2020). A survey on blocking technology of entity resolution. *Journal of Computer Science and Technology*, 35(4):769–793.

Li, H., Li, S., Hao, F., Zhang, C. J., Song, Y., and Chen, L. (2024). Booster: leveraging large language models for enhancing entity resolution. In *Companion Proceedings of the ACM Web Conference 2024*, pages 1043–1046.

Mestre, D. G., Pires, C. E. S., and Nascimento, D. C. (2017a). Towards the efficient parallelization of multi-pass adaptive blocking for entity matching. *Journal of Parallel and Distributed Computing*, 101:27–40.

Mestre, D. G., Pires, C. E. S., Nascimento, D. C., de Queiroz, A. R. M., Santos, V. B., and Araujo, T. B. (2017b). An efficient spark-based adaptive windowing for entity matching. *Journal of Systems and Software*, 128:1–10.

Nanayakkara, C., Christen, P., and Christen, V. (2025). Unsupervised evaluation of entity resolution. *ACM Journal of Data and Information Quality*, 17(1):1–31.

Nascimento, D. and Silva, V. (2025). Explorando uma nova métrica para calcular a precisão acurada de resultados de blocagem em tarefas de resolução de entidades. In *Anais do XL Simpósio Brasileiro de Bancos de Dados*, pages 15–27, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/sbbd.2025.246994.

Nascimento, D. C., Pires, C. E., and Mestre, D. (2016). Data quality monitoring of cloud databases based on data quality slas. In *Big-Data Analytics and Cloud Computing: Theory, Algorithms and Applications*, pages 3–20. Springer.

Nascimento, D. C., Pires, C. E. S., and Mestre, D. G. (2020a). Exploiting block co-occurrence to control block sizes for entity resolution. *Knowledge and Information Systems*, 62(1):359–400.

Nascimento, D. C., Pires, C. E. S., and Nobrega, T. P. (2020b). Configurable assembly of classification rules for enhancing entity resolution results. *Information Processing & Management*, 57(3):102224.

Nascimento, D. C., Pires, C. E. S., and Nóbrega, T. P. (2025). A generalized approach to perform unsupervised blocking key selection for entity resolution. *Information Sciences*, page 122952.

Papadakis, G., Koutrika, G., Palpanas, T., and Nejdl, W. (2013). Meta-blocking: Taking entity resolutionto the next level. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1946–1960.

Papadakis, G., Papastefanatos, G., and Koutrika, G. (2014). Supervised meta-blocking. *Proceedings of the VLDB Endowment*, 7(14):1929–1940.

Papadakis, G., Papastefanatos, G., Palpanas, T., and Koubarakis, M. (2016a). Scaling entity resolution to large, heterogeneous data with enhanced meta-blocking. In *EDBT*, pages 221–232.

Papadakis, G., Skoutas, D., Thanos, E., and Palpanas, T. (2020). Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys (CSUR)*, 53(2):1–42.

Papadakis, G., Svirsky, J., Gal, A., and Palpanas, T. (2016b). Comparative analysis of approximate blocking techniques for entity resolution. *Proceedings of the VLDB Endowment*, 9(9):684–695.

Zeakis, A., Papadakis, G., Skoutas, D., and Koubarakis, M. (2023). Pre-trained embeddings for entity resolution: an experimental analysis. *Proceedings of the VLDB Endowment*, 16(9):2225–2238.