Database Matching

Amir Bazzi and Sarah Sayyed

March 2024

Abstract

In the current era of extensive data generation, the ability to match databases accurately has become more vital than ever, especially given the heightened concerns surrounding data privacy. This paper delves into the complexities of database matching amidst the prevalent issue of random column deletions, which often complicate synchronization in time-indexed databases. We address this challenge by developing an innovative approach that not only enhances matching accuracy but also contributes to detecting deletion patterns. This work sets out to refine the theoretical understanding of achievable database growth rates, presenting strategies that capitalize on uniform deletion patterns without relying on shared deletion information among rows.

Moreover, we extend our exploration into the realm of Gaussian databases, introducing an algorithm tailored to detect correlations and facilitate alignment recovery. Our proposed method undergoes rigorous experimentation, validating its robustness against a variety of error types while respecting the theoretical error probability bounds.

Furthermore, we investigate the intricacies of error probabilities in alignment recovery, examining the impact of correlation coefficients on the reliability of the process. We provide comprehensive bounds on two error probabilities: the probability of erroneous recovery and the probability of undetected mismatches. The research confirms that the optimized tuning of key parameters, such as theta and beta, is crucial to the algorithm's effectiveness, thus influencing the overall data management and utilization in a multitude of applications. The findings of this study have significant implications for the fields of information theory and privacy, laying the groundwork for more secure and efficient database matching techniques in the future.

Contents

List of Figures

1 Chapter 1: Database Matching Under Column Deletions

1.1 Introduction

The introduction of the paper discusses the significant increase in data collection and privacy risks due to the proliferation of smart devices and social media. It highlights that anonymization alone is insufficient to prevent privacy leakage, as demonstrated by practical attacks on anonymized databases. The paper aims to advance the understanding of database matching under the condition of random column deletions, motivated by the need to address synchronization errors in time-indexed databases. By investigating the conditions for successful database matching, including the role of partial deletion information, the authors propose a novel approach to improve matching accuracy and develop a deletion detection algorithm, contributing to the field of information theory and privacy.

1. **Unlabeled Database**

   An $m \times n$ matrix $C = \{X_{ij} \in \mathcal{X}_{m\times n}\}$, with entries X_{ij} independently and identically distributed according to a discrete alphabet $\mathcal{X}$ with distribution p_X. Each row X_i corresponds to a user, where m and n signify the number of users and attributes, respectively.

2. **Column Deletion Pattern**

   Denoted as $\mathbf{D} = \{D_1, D_2, \ldots, D_n\}$, is a random vector with i.i.d. Bernoulli(δ) entries, independent of C. Here, δ represents the column deletion probability, and $D_i = 1$ indicates the deletion of the ith column.

3. **Column Deleted Labeled Database**

   For an unlabeled database $C^{(1)}$ and a column deletion pattern $\mathbf{D}$, along with a permutation Θ of $[m]$, the resultant $C^{(2)}$ represents the database after deletion and row permutation. The transformation is defined by $R(i^{(2)}) = E$ if $D_i = 1$, indicating deletion, and $\Theta \circ R(i^{(1)})$ otherwise. Here, $R(i^{(j)})$ denotes the ith column of $C^{(j)}$, with E representing an empty string for deleted columns.

4. **Deletion Detection Pattern**

   Represented as $\mathbf{A} = \{A_1, A_2, \ldots, A_n\}$, a random vector independent of $C^{(1)}$, with a conditional distribution $P(A_i = 1|D_i) = \alpha \cdot 1[D_i = 1]$, where α is the deletion detection probability, indicating the likelihood of correctly identifying a deleted column

5. **Database Growth Rate**

   Defined as $R = \lim_{n\to\infty} \frac{1}{n} \log_2 m$, represents how the size of the database scales relative to the number of attributes as n approaches infinity.

6. **Successful Matching Scheme**

Given a deletion detection pattern A_n, a matching scheme is a sequence of mappings $s_n : (C^{(1)}, C^{(2)}) \to \hat{\Theta}_n$ where $\hat{\Theta}_n \in [m]^m$ is the estimate of the correct permutation Θ_n. The scheme is successful if

$$P(\Theta_n(I) = \hat{\Theta}_n(I)) \to 1 \text{ as } n \to \infty.$$

Here, the index I is drawn uniformly from $[m]$. The dependence of $\hat{\Theta}_n$ on A_n is omitted for brevity.

1.2 Achievable database growth rate

Our objective is to outline achievable growth rates for databases, incorporating effective matching schemes. We introduce a distinctive strategy that initiates by removing all detected deleted columns from C(1), taking advantage of the uniform deletion pattern across rows. This is followed by applying a row-matching scheme, treating each row as separate entities and overlooking their identical deletion patterns. This approach is chosen to assess the strategy's efficiency independently of the shared deletion patterns among rows. Moreover, we highlight the prospect of implementing matching at the database level as a means to potentially enhance database growth rates. This discussion aims to showcase the depth of our strategic approach to optimizing database expansion without the direct exploitation of consistent deletion patterns across rows.

Consider an unlabeled database generated from a distribution p_X over an alphabet $\mathcal{X}$, subject to a column deletion probability δ, where $\delta < 1 - \frac{1}{|\mathcal{X}|}$. Given a deletion detection probability α, we can establish that any database growth rate defined by

$$R < [(1-\alpha\delta)H(\mathcal{X}) - H_b\left(\frac{1-\delta}{1-\alpha\delta}\right) - (1-\alpha)\delta\log(|\mathcal{X}|-1)]^+$$

is achievable. Here, H represents the entropy function, H_b the binary entropy function, and the notation $[.]^+$ denotes the positive part function. Importantly, it's feasible to reorganize the right-hand side of the inequality as:

$$[(1-\delta)H(\mathcal{X}) - (1-\alpha)\delta\left(\log(|\mathcal{X}|-1) - H(X)\right) - (1-\alpha\delta)H_b\left(\frac{1-\delta}{1-\alpha\delta}\right)]^+$$

This rearrangement highlights that the term $(1-\delta)H(\mathcal{X})$ corresponds to the achievable rate when full information on deletion locations is available (i.e., when $\alpha = 1$). In our work, we

endeavor to establish the proof where $\alpha = 1$ is applicable. The probability of error, denoted as P_e, can be bounded by the expression

$$\begin{aligned} P_e =& P(K < k, A < a, \text{matching error}) \\ &+ P(K < k, A > a, \text{matching error}) \\ &+ P(K \geq k, A < a, \text{matching error}) \\ &+ P(K \geq k, A > a, \text{matching error}) \end{aligned} \tag{1}$$

- Let K is defined as the total number of retained columns in a database after deletion, calculated by $K = n - \sum_{i=1}^{n} D_i$, where n is the total number of columns, and D_i indicates whether the i-th column is deleted ($D_i = 1$ for deletion). We choose a threshold $k = \lfloor n(1 - \delta - \tilde{\epsilon}) \rfloor$, with δ representing the column deletion probability and $\tilde{\epsilon}$ a small positive constant to adjust the threshold.

  By denoting the probability that K falls below this threshold k as κ_n, and applying the Law of Large Numbers, we conclude that κ_n converges to zero as the database size, or equivalently, n, grows infinitely large, symbolically, $\kappa_n \to 0$.

- Further, let I_A represent the set of indices for columns identified as deleted, and let $A = |I_A| = \sum_{i=1}^{n} A_i$, where A_i is a binary indicator of detection for the i-th column's deletion. Setting a detection threshold $a = \lfloor (n - k)(\alpha - \hat{\epsilon}) \rfloor$—with α being the deletion detection probability and $\hat{\epsilon}$ a small positive constant for adjustment—we observe that for any count of detected deletions A meeting or exceeding a, the system performs as intended.

  The probability that A is less than a, denoted by μ_n, also tends to zero as n approaches infinity, adhering to the Law of Large Numbers. This is expressed mathematically as $\mu_n \to 0$ for $n \to \infty$, indicating that the probability of underestimating the number of detected deletions becomes negligible in large databases.

Let $F(n, k, |\mathcal{X}|)$ denote the number of $|\mathcal{X}|$-ary sequences of length n that contain at least one fixed $|\mathcal{X}|$-ary sequence of length k. Mathematically, $F(n, k, |\mathcal{X}|)$ is defined as:

$$F(n, k, |\mathcal{X}|) = |\{x^n : \exists\, i_1 < i_2 < \ldots < i_k \text{ such that } x_{i_l} = \tilde{x}_l \,\forall l \in \{1, \ldots, k\}\}|$$

where x^n represents an $|\mathcal{X}|$-ary sequence of length n, and $\tilde{x}^k$ is a specific $|\mathcal{X}|$-ary sequence of length k embedded within x^n. The indices $i_1, i_2, \ldots, i_k$ indicate the positions in x^n where the sequence $\tilde{x}^k$ appears, ensuring that each element x_{i_l} of the sequence x^n matches the corresponding element $\tilde{x}_l$ of the sequence $\tilde{x}^k$ for all positions l within the subsequence.

We define $\tilde{T}$ as the set of tuples $(t_1, \ldots, t_k)$, where each t_i is a member of the set $\{1, \ldots, n\}$, such that for the sequence X, the subsequence formed by $X(t_1), \ldots, X(t_k)$ matches the specific sequence $\tilde{x}(1), \ldots, \tilde{x}(k)$. In formal notation, $\tilde{T} = \{(t_1, \ldots, t_k) \mid t_i \in \{1, \ldots, n\} \text{ for all } i, \text{ and } X(t_1, \ldots, t_k) = \tilde{x}(1, \ldots, k)\}$.

We define the watching error ϵ_{1j} as the collision with row 1 such that another row with $\epsilon^{(1)}$ will match with $\epsilon^{(2)}$. The probability of a matching error given $K \geq k$ and $A \geq a$ is expressed as $\Pr[\text{matching error}/K \geq k, A \geq a] = \Pr[\epsilon_{1j}/K \geq k, A \geq a]$.

The probability distribution $p(x^n)$, where x^n is a sequence of length n, is $\frac{k}{n}$ times a Bernoulli distribution with parameter $\frac{k}{n}$. Since the rows are independent and identically distributed (i.i.d.), $\forall x^n \in A_\epsilon^{(n)}\left(\frac{k}{n}\right)$.

We define $|L| = F(n,k,|X|) \leq 2nH_b\left(\frac{k}{n}\right) \leq 2^{(nH_b(k/n)} \cdot (|\mathcal{X}|-1)^{n-k}$ and let $T(y,I_A) = \{x \in X^n | x([n] \setminus I_A) \in A_\epsilon^{(n-a)} \text{ contains } y\}$, and y be the row of $C(2)$ matching with the row X_1 of $C(1)$. It is clear that $|T(y,I_A)| \leq F(n-a,k,|\mathcal{X}|)$, since $|T(y,I_A)| \subseteq |L|$.

$\forall x \in T(y,I_A) : x^{(n-a)}$ i.i.d. $\sim p_x$, thus $T(y,I_A) \in A_\epsilon^{(n-a)} \Rightarrow p_x \leq 2^{-(n-a)(H(X)-\epsilon)}$.

The matching error occurs when $x_j^{(n-a)} \in T(y,I_A)$, so

$$\begin{aligned} P_{\epsilon_{ij}} &= P[X_j \in T(y,I_A)] \\ &= \sum_{x \in T(y,I_A)} p(x) \\ &\leq 2^{-(n-a)}(H(X)-\epsilon)F(n-a,k,|\mathcal{X}|) \\ &\leq (n-a)2^{-(n-a)}(H(X)-\epsilon-H_b(k/n-a))(|\mathcal{X}|-1)^{n-a-k} \end{aligned}$$

$$P[\text{matching error}] = \sum_{j=2}^{2^{nR}} P_{\epsilon_{ij}} \leq (n-a)2^{-n[(1-\frac{a}{n})(H(X)-\epsilon-H_b(\frac{k}{n-a}))-R]}(|\mathcal{X}|-1)^{n-a-k}$$

$$\begin{aligned} P_e \leq\ & \kappa_n \cdot \mu_n P[\text{matching error}/K<k, A<a] \\ & + \kappa_n \cdot (1-\mu_n) P[\text{matching error}/K<k, A \geq a] \\ & + (1-\kappa_n) \cdot \mu_n P[\text{matching error}/K \geq k, A<a] \\ & + (1-\kappa_n) \cdot (1-\mu_n) P[\text{matching error}/K \geq k, A \geq a] \end{aligned}$$

As $n \to \infty$ if

$$R < \left(1-\frac{a}{n}\right)(H(X)-\epsilon-H_b)\left(\frac{k}{n-a}\right) - \left(1-\frac{a}{n-k}\right)\frac{n-k}{n}\log\left(|\mathcal{X}|-1\right)^+$$

Thus, we can argue that any rate R satisfying

$$R < (1-\alpha\delta)\,H(X) - H_b\left(\frac{1-\delta}{1-\alpha\delta}\right) - (1-\alpha)\delta\log\left(|\mathcal{X}|-1\right)^+$$

2 Database Correlation Detection and Alignment Recovery of Gaussian Databases

2.1 Introduction

The exponential growth of data and the pervasive use of databases across various sectors such as banking, web applications, telecommunications, and machine learning underscore their importance. However, in many instances, databases need to be correlated to effectively manage and utilize the data they contain. Therefore, the detection of database correlations has emerged as a critical aspect for ensuring seamless data integration, analysis, and cleansing. This chapter aims to develop and implement an algorithm for detecting database correlations and aligning recovery processes. Furthermore, it seeks to examine error probabilities and establish their bounds within this context.

2.2 Description

The algorithm primarily addresses two key issues: correlation detection and alignment recovery. Correlation detection involves a hypothesis testing approach, where the null hypothesis assumes independence between the two databases, while the alternate hypothesis suggests correlation with unknown permutation $\sigma \in S_n$ and known correlation coefficient $\rho > 0$. Upon accepting the latter hypothesis, the algorithm proceeds to the next phase, alignment recovery, which involves estimating the permutation.

The algorithm's input: two gaussian databases X^n and Y^n with n rows and d columns.

Correlation detection: Testing of the following two hypothesis

$$H_0 : (X_1, Y_1), \ldots, (X_n, Y_n) \overset{\text{IID}}{\sim} \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$$

$$H_1 : (X_1, Y_{\sigma(1)}), \ldots, (X_n, Y_{\sigma(n)}) \overset{\text{IID}}{\sim} \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & \rho\\\rho & 1\end{bmatrix}\right)$$

Alignment recovery: If H_1 is accepted the algorithm must output $\psi(X_n, Y_n)$(list of all probable matched pairs that must not be permutations).

2.3 Threshold-and-Clean Algorithm

2.3.1 Normalize the row vectors

$$\tilde{X}_i = \frac{X_i}{\|X_i\|}, \quad i \in \{1, 2, ..., n\}$$
$$\tilde{Y}_j = \frac{Y_j}{\|Y_j\|}, \quad j \in \{1, 2, ..., n\}$$

2.3.2 Define the statistic

For $\theta \in (0, 1)$,

$$N(\theta) \triangleq \sum_{i=1}^{n} \sum_{j=1}^{n} 1\{\tilde{X}_i^T \tilde{Y}_j \geq \theta\}$$

2.3.3 Detection phase

$$\phi_N(\tilde{X}, \tilde{Y}) = \begin{cases} 0 & N(\theta) < \beta nP, \quad H_0 \text{ is accepted} \\ 1 & N(\theta) \geq \beta nP, \quad H_1 \text{ is accepted} \end{cases}$$

2.3.4 Alignment Recovery

The (i, j) pairs for which $\tilde{X}_i^T \tilde{Y}_j \geq \theta$ are stored in a list and then this list is cleaned if there are permutation in a way that for example if both tuples $(i1, j1)$ and $(i1, j2)$ exists in the list we remove the one with least correlation.

2.4 Experiments

Beyond the theoretical aspect of this algorithm, we have implemented it using python code in order to test its reliability against the various types of errors (type-I, type-II, alignment errors..)and to show whether the theoretical bounds of the probabilities of these are well respected.

2.4.1 Algorithm implementation

We designed the implementation of the algorithm as a function that takes the two databases, and the parameters θ and β. Then calculate the inner product matrix between them, and locate the row pairs having inner product greater than θ. If the number of these pairs is greater than the threshold specified in 2.3.3 then the algorithm states that these two databases are correlated. Further, it cleans these pairs in a way that each row in the first database is aligned with only one row in the second database.

Figure1 and Figure2 are graphical representations of the results of this algorithm tested on a databases of size:(number of rows=20, number of columns=5) before and after cleaning the aligned pairs.

Note that in the second figure each row in the first database is aligned exactly with one row in the second database. The number of uncleaned pairs: 102 The number of cleaned pairs: 20

2.5 Type I and Type II probability of error

Before commencing the alignment process, the algorithm must decide between two hypotheses: the first posits that the two databases are independent, while the second suggests they are correlated. The Type I error probability refers to the likelihood of accepting the alternative hypothesis when the null hypothesis is true. Conversely, the Type II error probability

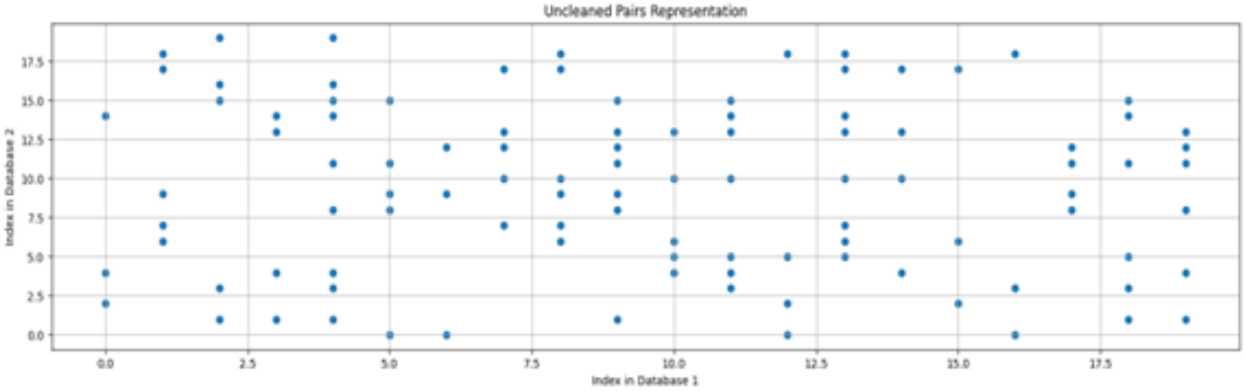

Figure 1: The resulted matched pair before cleaning

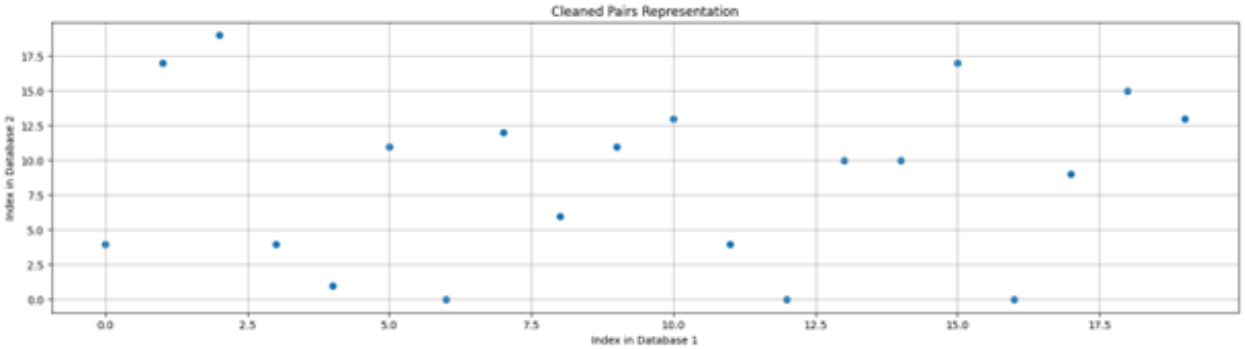

Figure 2: The resulted matched pair after cleaning

indicates the likelihood of accepting the null hypothesis when the alternative hypothesis is true. We incorporate these error probabilities into our algorithm. The graphs in figures 3 and 4 illustrate how these probabilities vary with the parameters β. Note that the size of databases used is (200×50).

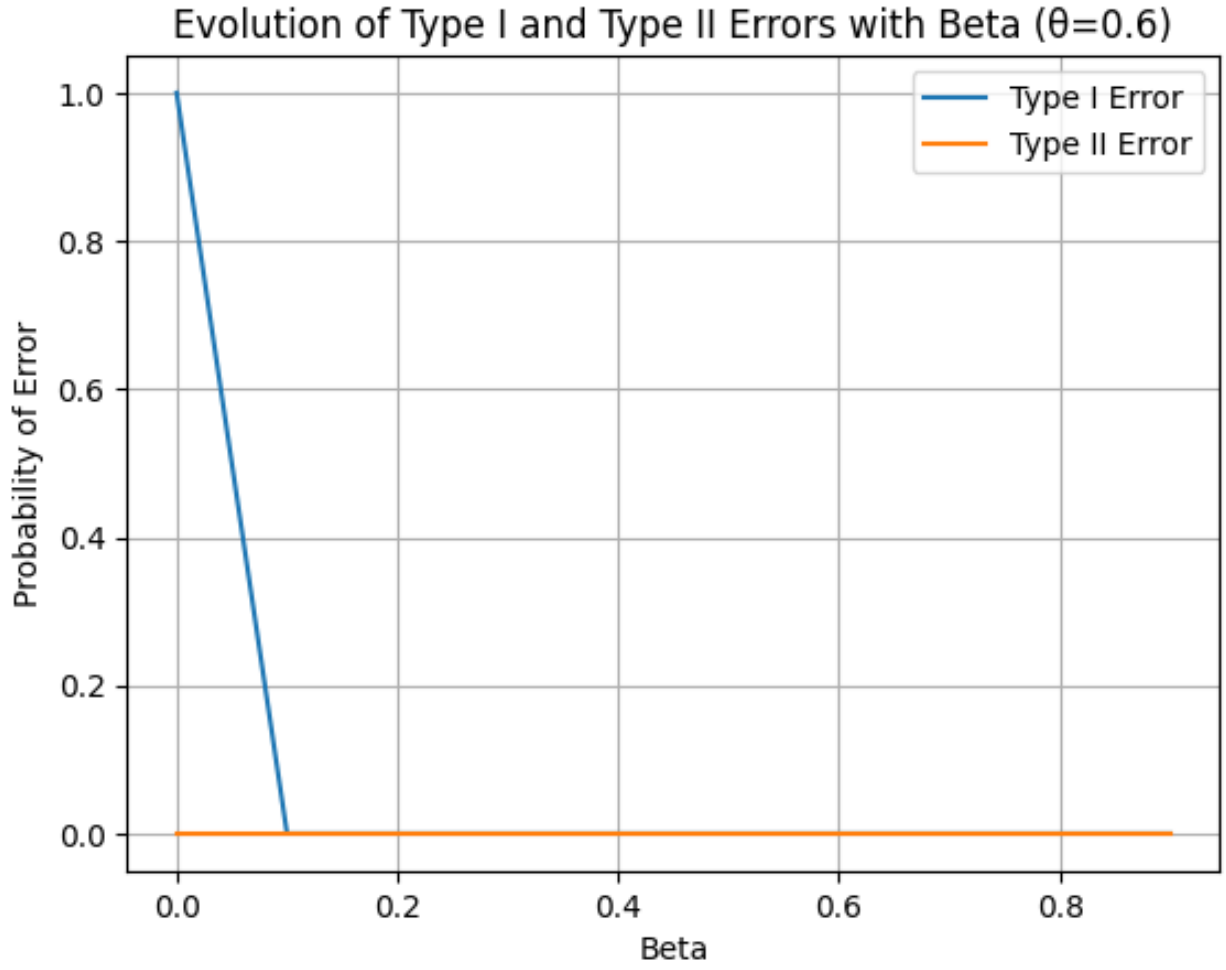

Figure 3: The evolution of type I and type II probabilities of error as a function of beta values

As we can see in Figure 3, when β is relatively low, then relatively few "dots" are required to accept H_1, hence the type-I error probability is high and the type-II error probability is low.

One of our main goals is to find a compromise between these two probabilities of errors. As depicted in Figure 4, for $\theta = 0.6$ and $\beta = 0.4$, there is a notable trade-off between these two error probabilities. Thus, we will mainly use these values in the rest of the experiments. Furthermore, we have observed the evolution of the probabilities of the two types of errors as a function of database size (rows and columns), as shown in figures 5 and 6.

We can conclude that as the number of columns increases, this algorithm becomes more robust to type I and type II errors.

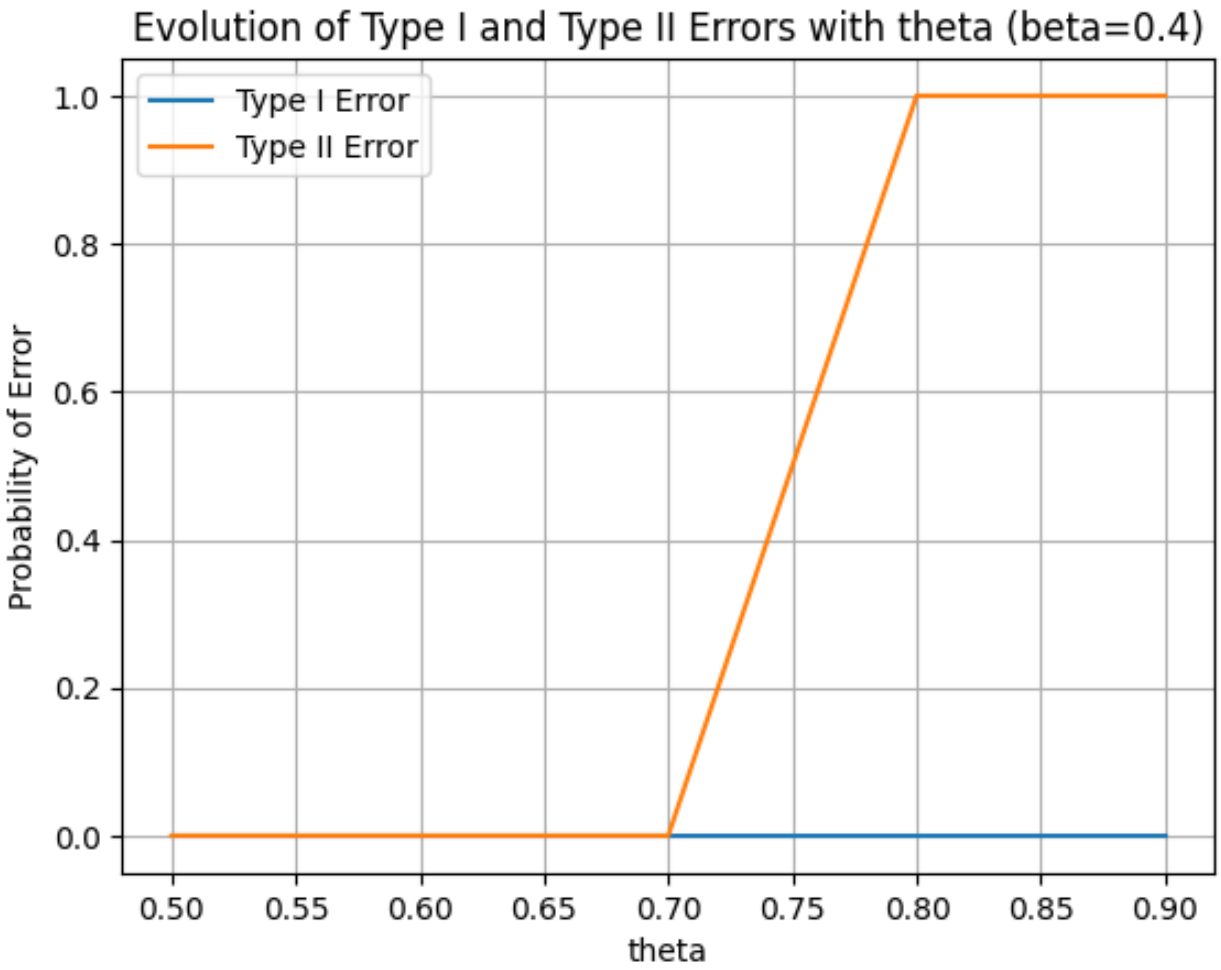

Figure 4: The evolution of type I and type II probabilities of error as a function of theta values

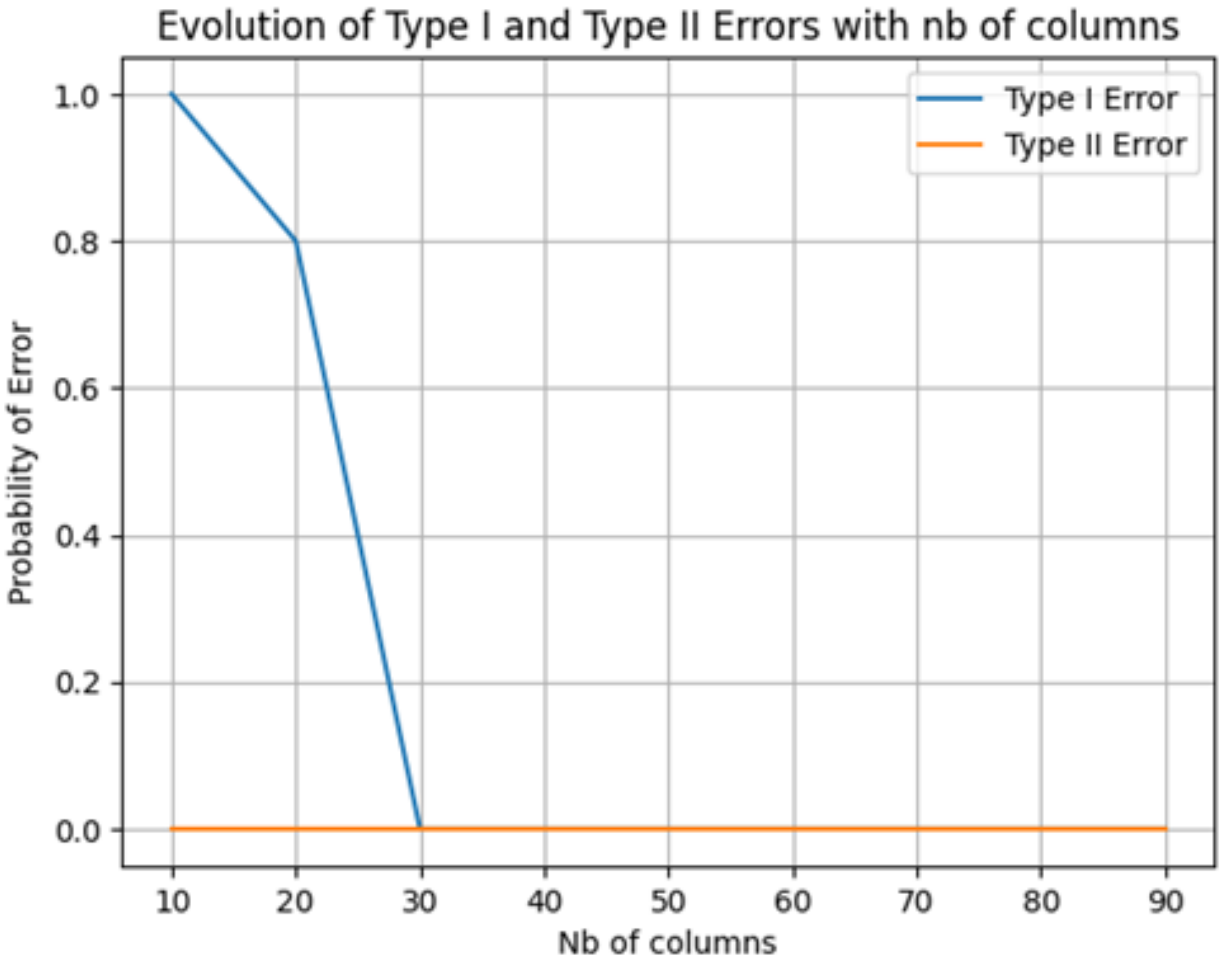

Figure 5: The evolution of type I and type II errors as a function of the number of columns.

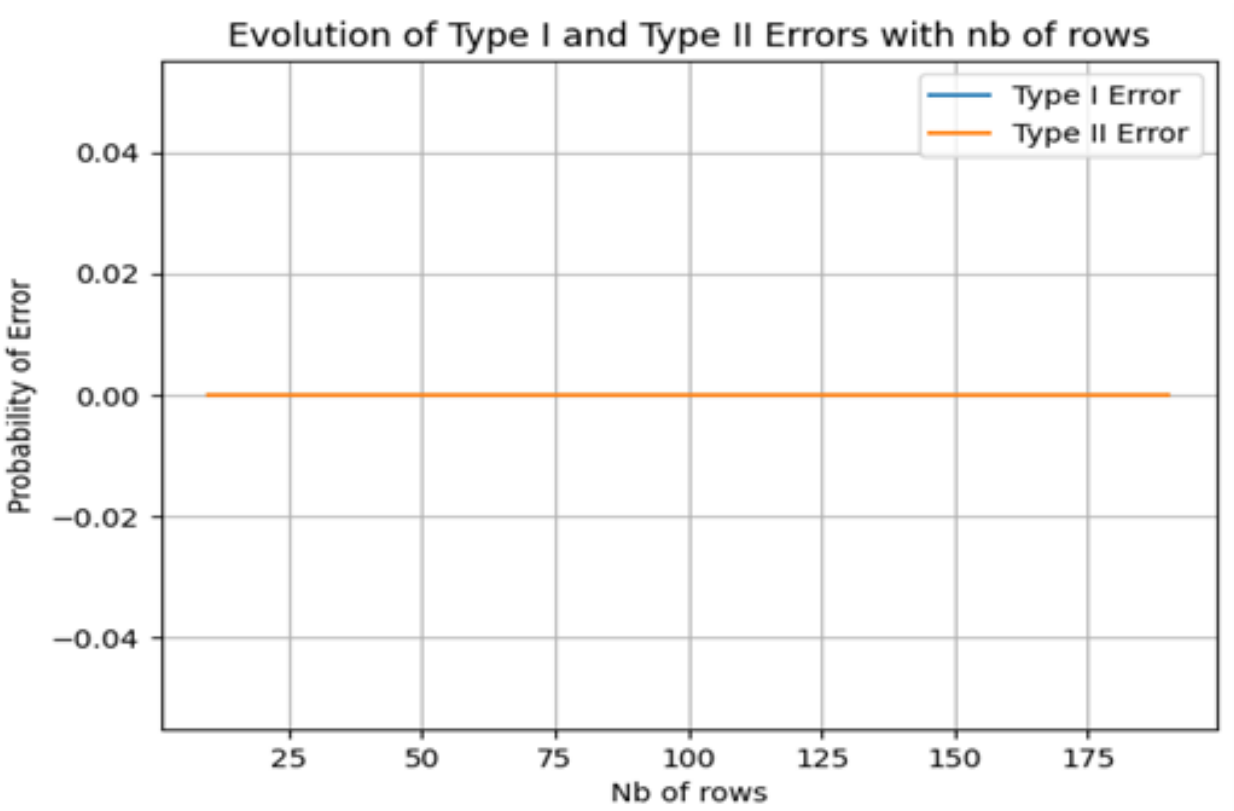

Figure 6: The evolution of type I and type II errors as a function of the row size.

The theoretical bounds on these probability of errors are the following: For the type-I error probability:

$$P_{\mathrm{FA}}(\phi_N) \leq \inf_{k\in\mathbb{N}} \left\{k(k+1)B(k)\left((n^2Q)^k \cdot \mathbb{I}\{n^2Q \geq 1\} + n^2Q \cdot \mathbb{I}\{n^2Q < 1\}\right)\right\} \left(\frac{1}{(\beta nP)^k}\right)$$

For the type-II error probability:

$$P_{\mathrm{MD}}(\phi_N) \leq \exp\left\{-\min\left(\frac{(1-\beta)^2 nP}{16nQ+2}, \frac{(1-\beta)n}{12}\right)\right\}$$

Where:

$$\begin{aligned} P(d,\rho,\theta) &\triangleq \mathbb{P}\left\{\widetilde{X}^T\widetilde{Y} \geq \theta\right\}, \\ Q(d,\theta) &\triangleq \mathbb{P}\left\{\widetilde{X}^T\widetilde{Y} \geq \theta\right\}, \\ B(k) &= \sum_{d=1}^{k} S(k,d) \cdot \frac{k^{2d}}{d!}. \end{aligned}$$

Where $S(k,\ell)$ is the Stirling number of the second kind, given by:

$$S(k,\ell) = \frac{1}{\ell!}\sum_{i=0}^{\ell}(-1)^i \binom{\ell}{i}(\ell-i)^k,$$

The result of the implementation shows that the first bound is equal to $8.32054710678649 \times 10^{-15}$ and the second equal to $4.5399929762484854 \times 10^{-5}$ for $\theta = 0.6$ and $\beta = 0.4$. Which are firmly respected.

2.5.1 Error probabilities of alignment recovery

Regarding the alignment recovery process, two kinds of errors may arise: the probability of error that the resulting recovery is not the correct permutation (defined as pe_1) and the probability of error that at least one pair of the recovery is a mismatch (defined as pe_2).

The theoretical bounds on these error probabilities are the following: **Theorem 2 (Permutation recovery probability bounds).** Let $n, d \in \mathbb{N}$ and $\rho, \theta \in (0,1]$ be given. Let $P = P(d,\rho,\theta)$ and $Q = Q(d,\theta)$ as defined above.

1. Regarding the erroneous recovery error probability in (7),

$$Pe_{,1}(\psi_{\mathrm{TC}}) \leq \min\left[1, n(1-P) + n(n-1)Q\right], \tag{2}$$

and,

$$Pe_{,1}(\psi_{\mathrm{TC}}) \geq \frac{n(1-P) + n(n-1)Q}{\max\{P, 1-Q\} + n(1-P) + n(n-1)Q}. \tag{3}$$

2. Regarding the mismatch-undetected error probability in (8),

$$Pe_{,2}(\psi_{\mathrm{TC}}) \leq \min\left[1, \frac{n(n-1)Q(1-P)^2(1-Q)^2}{n^4}\right]. \tag{4}$$

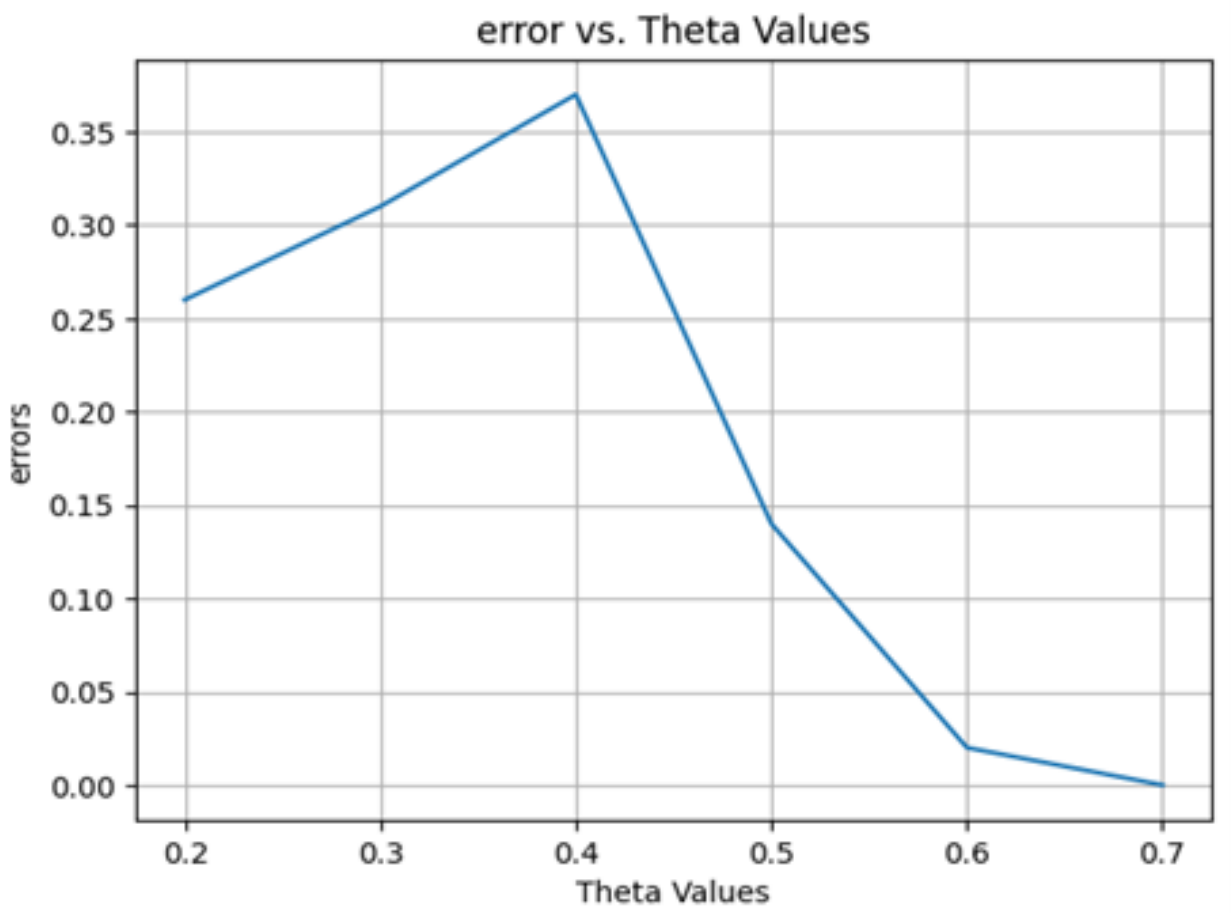

Figure 7: The evolution of P_{e2} error probability with theta values

We plot in figure 7 the evolution of pe_2 with the values of θ for $n = 200$, $d = 50$ for databases with correlation coefficient = 0.7.

Additionally, we observed the evolution of the normalized average output size over 50 databases with $n = 200$, $d = 50$ and correlation coefficient = 0.7 as a function of θ values in figure 8.

2.5.2 Analysis on Theta's Impact

We can observe that for both small and large values of theta, the reliability of our database matching algorithm is high, attributable to the enhanced accuracy in identifying correlated rows across databases. Specifically, when theta values are small, a larger number of correlated rows surpass the set threshold, alongside some independent rows. However, our cleaning process effectively eliminates these independent rows, thereby ensuring high reliability.

Conversely, for larger theta values, predominantly the rows with maximum correlation are identified, which significantly boosts the reliability of our matching process. Nonetheless, the scenario becomes intricate for intermediate theta values. In such cases, while correlated rows continue to exceed the threshold, the likelihood of independent pairs doing the same escalates, occasionally evading the cleaning process and thereby, potentially introducing errors.

2.5.3 Observations on Output Size

Additionally, our investigation into the evolution of the normalized average output size across 50 databases, each with 200 users (n=200) and 50 attributes (d=50), and a correlation coefficient of 0.7, yields insightful patterns. Particularly, figure 8 demonstrates that at relatively lower theta values, the output size tends to be higher. This phenomenon can be attributed to the fact that both correlated and uncorrelated pairs exceed the bound, leading to an inflated output size. However, as theta increases, surpassing the bound becomes increasingly challenging, resulting in a reduction in output size.

2.5.4 Optimal Theta for Reliability and Output Size

From figures 7 and 8, it becomes evident that the optimal trade-off between reliability and output size is achieved at a theta value of 0.6, particularly when the correlation coefficient between the two databases stands at 0.7. This balance is crucial for maximizing the efficiency of our database matching algorithm while minimizing potential errors.

3 Conclusion

With the continuous increase in data volume, database matching has become a crucial step in numerous applications. The proposed algorithm functions as a Gaussian database correlation detector and matcher. It offers simplicity and high efficiency. However, its effectiveness heavily relies on the threshold parameters (theta and beta). The optimal tuning of these parameters is contingent upon the specific application and its objectives.

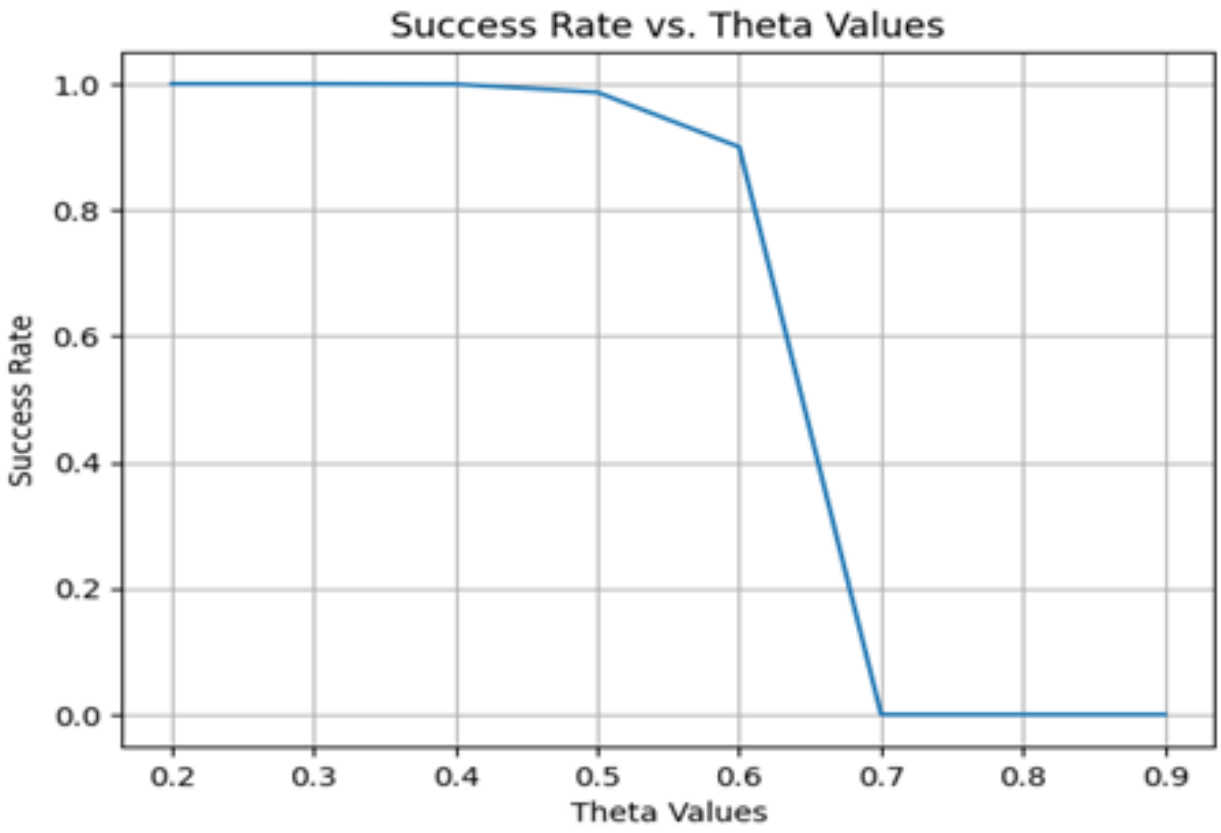

Figure 8: The evolution of the output size as a function of theta values