Evaluating String-Based Similarity Algorithms for Duplicate Record Identification in Databases

Mohammed Nasser Salem [1]
Mohammed Fadhl Abdullah[2*]

Received: 03/01/2026
Revised: 06/01/2026
Accepted: 15/01/2026

[1]Department, Faculty of Engineering, Aden University, Aden, Yemen
[2]Department, Faculty of Engineering and Computation, University of Science and Technology, Aden, Yemen.
* Corresponding author Email: m.albadwi@ust.edu

Evaluating String-Based Similarity Algorithms for Duplicate Record Identification in Databases

Mohammed Nasser Salem[1,2], Mohammed Fadhl Abdullah[1,3]
[1]College of Engineering and Computing, University of Science and Technology, Aden, Yemen
[2] kaio.nasser@gmail.com,[3]m.albadwi@ust.edu

Abstract— Duplicate records negatively affect the accuracy, reliability, and efficiency of database systems. This paper presents a comparative study of two widely used string-based similarity algorithms: Simil and Jaro–Winkler. Both algorithms measure textual similarity between records in order to identify entries that refer to the same real-world entity. The findings show that Simil is more operative for multi-word fields such as names and addresses, while Jaro–Winkler performs better for short words and typographical errors. The study highlights the strengths and limitations of both algorithms and provides practical guidance for their use in database duplicate detection systems. The study is helpful for academic interest and systems developers to get ideas about how to deal with unnecessary data and to improve the overall data quality.

Keywords— *Database, Duplicate Records, Jaro-Winkler, Simil algorithm, Data cleansing.*

I. Introduction

Databases often contain duplicate records that refer to the same real-world data. These duplicates reduce the quality of the data and take up unnecessary storage space. As databases grow, the chance of having repeated or badly recorded entries increases, especially when information is entered by hand. Mistakes such as misspellings, missing values, and inconsistent formats make it difficult to identify records that represent the same entity.

Duplicate detection is different from traditional information retrieval, where similarity focuses on meaning. Instead, duplicate detection looks for records that are almost the same in wording or structure. Because many records do not share a common unique key, matching them becomes a challenging task.

This article is about small topics in matching records, spell-checker algorithms, and data preparation before error detection. It also presents two algorithms to identify similarity in string-based data in records that have duplicated data. And brief basic concepts about data cleaning—all that helps researchers and systems development to improve data accuracy and saves storage space for databases.

II. LITERATURE REVIEW

The importance of databases in modern technology stems from the existence of fundamental references that can be relied upon to store personal, social, or operational data needed by any data-based system. Therefore, as work increases and more data is stored in the database, the likelihood of data duplication in fields also increases, so there is a need to find a way to detect the duplication and eliminate it from tables in the database, so there are a lot of algorithms that are designed to detect text similarity in database tables.

Problem of matching records in database tables: In some contexts two records are considered equivalent if they are semantically equal, meaning they refer to the same real-world entity, a relationship that satisfies reflexivity, symmetry, and transitivity. Record-matching algorithms attempt to determine this semantic equivalence using syntactic comparisons of record data. Because these syntactic measures only approximate true semantic meaning, errors are unavoidable and exact equivalence may not always be identified. Nevertheless, such approximations are generally effective, producing relatively few errors and yielding acceptable results in practice. [1]

Effective duplicate detection requires careful selection of matching fields and similarity metrics and domain knowledge to define matching rules [2].

According to the database and the type of data, there is a hybrid technique that combines the field comparison (string matching, phonetic matching, and edit distance) and the rule-based heuristics to handle partial or noisy duplicates. Clustering algorithms to group similar records for efficient comparison. [2]

In English, spell-checker algorithms are generally divided into two main groups:

A. Phonetic Algorithms:

These algorithms compare words based on how they sound when pronounced. They try to catch errors where different spellings have similar sounds. For example, in English [3], “wr” and “r” sound alike (as in wrong and right), and “u” and “oo” can sound similar (as in Luther and loop).

Because of this, phonetic algorithms help identify misspellings that still sound correct. [3]

B. Approximate String-Matching Algorithms:

These algorithms measure how similar two strings are by calculating a similarity score between 0 and 1: the value 1 means the strings are exactly the same. -The value 0 means they share no similarity.
The score is based on factors such as the number of matching letters or repeated characters, the order and position of those letters, and other structural similarities. [3]
Data preparation: The data preparation before the error detection stage includes a parsing of data transformation and a standardization step. These steps improve the quality of the inflow data and make the data comparable and more usable. [4]

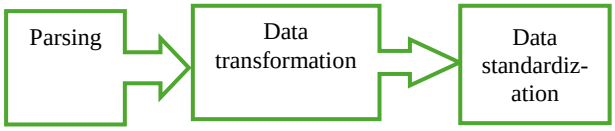

Fig1. Steps in data preparation

The first step of the data preparation is to identify, locate, and then isolate the individual data elements. Here, each long string is divided into smaller pieces. Separate the names from the field of address and so on. This step is followed by the process of data conversion to ensure the required data types. The focus here is on the individual fields without considering relationships with other fields.[4]
Data standardization is the process of converting information in specific fields into a uniform and consistent format. It is essential for data that may be represented in multiple ways across different sources and must be standardized before duplicate detection begins. Without standardization, records containing the same identifying information may be incorrectly treated as non-duplicates, with address data being one of the most common examples requiring this process.[4]
After data preparation, records are stored in tables with comparable fields, and the relevant fields for comparison must be carefully selected, as comparing unrelated fields (such as last name and address) is meaningless. However, even with proper parsing, standardization, and field selection, duplicate record matching remains challenging because misspellings and varying data entry conventions can still produce multiple representations of the same real-world entity. [4]
There is important logically steps To remove duplicated similarity records in database so after data preparing first you have to determine the type of data that you want to detect the similarity in it , the second step need to exam the data by appropriate similarity detected algorithm in final step you need to eliminate the unwanted similar using program that designed for that particular purpose.

String-Based Similarity Metrics: are designed to handle typographical errors well also used to detect duplicated text data in database tables, such: Edit distance, Affine gap distance, Smith-Waterman distance, Jaro distance metric, Jaro–Winkler distance, Q-gram distance. Simil String based, Levensthein distance...etc. [5]
In this article, the focus is on two well-known approximate string-matching algorithms: SIMIL and Jaro–Winkler.

III. DATA PREPARATION FOR DUPLICATE DETECTION

Before applying similarity algorithms, data must be prepared through parsing, transformation, and standardization. These steps improve data consistency and comparability.
Parsing separates composite fields into atomic elements (e.g., splitting full names into first and last names).
Transformation converts values into appropriate data types.
Standardization ensures that values follow a uniform format (e.g., address abbreviations, date formats).
Without proper preparation, even identical entities may not be detected as duplicates.

IV. STRING-BASED SIMILARITY ALGORITHMS

This study focuses on two approximate string-matching algorithms: Simil and Jaro–Winkler.

A. Simil Algorithm:

The algorithm is based on checking the similarity between two strings, S1 and S2, as original strings, through steps of dividing the text into left and right sides and extracting the Longest Common Substring (LCS); that operation continues until no LCS remains. Finally, to find the similarity factor between them, divide the total length of the LCS over the length.
The similarity score is defined as:

$$Simil(S_1, S_2) = \frac{\sum LCS}{\max(|S_1|, |S_2|)}$$

where:

- S1 and S2 are the two strings,
- LCS is the longest common substring at each iteration.

The score ranges between 0 and 1.
So, we take example for two words Pennsylvania and Pencilvaneya to compare using Simil Algorithm [6].

Table 1. How Simil Algorithm works on strings.

Word1	*Word2*	*Common substring*	*Length*
Pennsylvania	*Pencilvaneya*	*Lvan*	*8*
Pennsy ia	*Penci eya*	*Pen*	*6*
nsy ia	*ci ey*	*A*	*2*
nsy i	*ci ey*	*(none)*	*0*
Subtotal			*16*
Length of original			*24*
Simil factor 16/24			*0.67*

Simil factor is value between 0 and 1. so the simil score is 0.67.

Jaro-Winkler Algorithm:
Jaro was introduced by Matthew A. Jaro as a comparator to be used in censuses and health data files, and later William E. Winkler improved it. [3].
To calculate the Jaro similarity we use formula:

Where:
- s1 and s2 are the lengths of the two strings
- m is number of matching characters
- t is number of transpositions.

And the winkler additional prefixes formula is:
JW=J+l.p.(1-j)
Where:
- l = length of common prefix (up to a max, usually 4)
- p is a scaling factor (usually 0.1).),

Consider the example of comparing two words; namely, Martha and Marhta using Jaro-winkler Algorithm.

Table 2. How Jaro–Winkler Algorithm works on strings.

	[1]	*[2]*	*[3]*	*[4]*	*[5]*	*[6]*
S_1	*M*	*a*	*r*	*t*	*h*	*a*
S_2	*M*	*a*	*r*	*h*	*t*	*a*

S1= 6 and S2 = 6 and the Compare T ↔ H → swapped positions → 1 transposition pair Number of transpositions (t) = 0.5

$$J = \frac{1}{3}\left(\frac{6}{6} + \frac{6}{6} + \frac{6 - 0.5}{6}\right)$$

$$\boldsymbol{J = \frac{2.9167}{3} = 0.9722}$$

The Jaro-score is: 0.9722. So, after add Winkler improvement:
- Common prefix: both start with MAR, l=3
- Use p=0.1 (standard)
- Compute 1-J=1-35/36= 1/36
- Compute the prefix term:

$$\boldsymbol{l.p.(1-j) = 3.0.1.\frac{1}{36} = \frac{3}{10}.\frac{1}{36}}$$
$$\boldsymbol{= 0.00833333}$$

- Add to Jaro: $\boldsymbol{JW = \frac{35}{36} + \frac{1}{120} = \frac{350}{360} + \frac{3}{360}} =$ 0.980556

So, the Jaro-winkler score is: 0.9805556. Jaro-winkler algorithm is perform better measurement of lexical similarity and suitable for short string data.[7].

V. COMPARATIVE EXAMPLE USING IDENTICAL STRINGS

To provide a fair comparison, both algorithms are applied to the same string pairs.
Example 1:
Pennsylvania vs. Pencilvaneya

Table 3. Working the algorithms works on Pennsylvania, and Pencilvaneya strings.

Algorithm	**Similarity Score**
Simil	0.67
Jaro–Winkler	0.84

Simil captures long shared substrings, while Jaro–Winkler benefits from character-level matching (see Table 3).

Example 2:
Martha vs. Marhta

Table 4. Working the algorithms works on Martha, and Marhta strings.

Algorithm	**Similarity Score**
Simil	0.72 (approx.)
Jaro–Winkler	0.98

Jaro–Winkler performs better on short strings with transposition errors (see Table 4).
Modern duplicate detection systems employ a range of algorithmic techniques to enhance processing efficiency. All these systems need to be adaptive to help identify the duplication patterns and to refine their detection strategies over time automatically. The estimation of the expected error rates in data cleansing processes is the main challenge. Where deriving the structured data from noisy sources becomes necessary to ensure data quality.
At this stage, it is important to introduce a brief overview of data cleaning, as it provides essential background knowledge that may be required in subsequent discussions.

Data cleansing is the process of improving the quality through correcting errors and inaccuracies (misspellings, missing values, or invalid entries) in data.

A. Single-Source Problems

The pre-defined schemas and integrity constraints in the single-source data ensure its quality. But, with a flat file (without schemas), unrestricted data entry is allowed, thereby increasing the possibility of inconsistencies. The responsibility of the database management system (DBMS) is to mitigate these issues. However, errors such as typographical errors cannot be entirely prevented through schema enforcement alone.

B. Multi-Source Problems

Qualitative error detection techniques are generally guided by three fundamental questions: what errors should be detected, how detection should be performed, and where detection should occur.

1. Error Type: Error detection methods are classified according to the types of identified errors. They may depend on the integrity constraints, such as functional dependencies and denial constraints. The duplicate records can be interpreted as key constraints.
2. Automation: Fully automated techniques, such as those identifying violations of functional dependencies, operate without user intervention. Other methods incorporate human judgment, for example, by requiring manual review or validation of potential duplicate records, thereby combining automated detection with expert oversight.
3. Business Intelligence Layer: Errors can occur at any stage of a business intelligence stack and are often propagated from source databases through the data processing pipeline. While many detection techniques focus on identifying errors at the source, some errors can only be detected at later stages when additional semantics and business logic are applied, such as enforcing budget constraints after aggregating costs and expenses. [9]

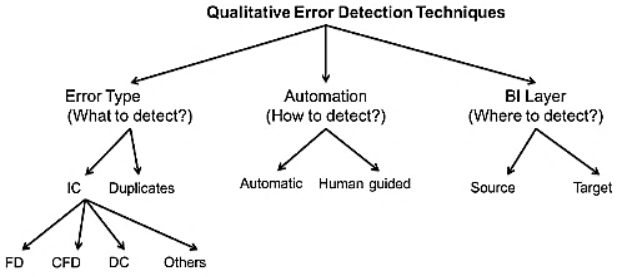

Fig2: Qualitative error detection techniques

Error Repairing: there are three important questions that every technique has to posed: the first one “What to Repair?”, then “How to Repair?”, and finally “Where to Repair?”. In the following, we classify the techniques on these axes, and discuss the impact on the design and efficiency of the techniques.[9]

Repair Target (What to Repair?): Data repair algorithms differ in their assumptions about data and quality rules. Some trust declared integrity constraints and modify only the data, others trust the data and allow constraints to be relaxed to handle issues like schema evolution, while a third group considers changing both data and constraints. Among methods that trust the rules, some focus on repairing a single error type at a time, whereas newer holistic approaches address multiple interacting error types simultaneously for more comprehensive data repair.[9]

Automation (How to Repair?): Repair approaches can be classified based on the tools used and the degree of human involvement in the repairing process. Some techniques are fully automatic, modifying the database to minimize the difference between the original and repaired versions according to a cost function. Other approaches involve humans to verify or suggest repairs, or to train machine learning models that support automated repair decisions.[9]

Repair Model (Where to Repair?): Proposed approaches can be classified by whether they repair the database in-situ or use an external model to represent possible repairs. Most techniques perform in-situ repairs, directly modifying the database, while non–in-situ approaches build models that capture multiple repair alternatives and support query answering through probabilistic or sampling-based methods. These techniques are organized using a taxonomy, with representative examples from each category discussed in detail.[9]

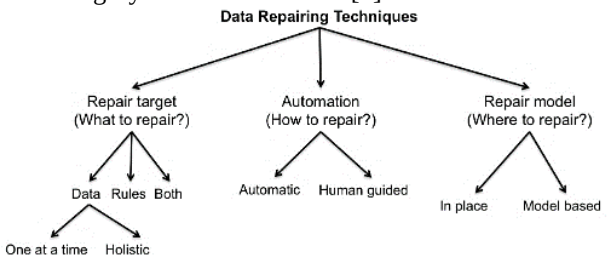

Fig 3: Error repairing techniques.

VI. DISCUSSION

At the end we notice that Simil and Jaro–Winkler, both measure textual similarity. Simil is well-suited for: (i) multi-word fields (names, addresses), (ii) token reordering, or (iii) long shared segments. While Jaro–Winkler is well-suited for: (i) short strings, (ii) typographical errors, or (iii) spelling variations.

Limitations:

Table 5 shows the mean limitations of these algorithms. Neither algorithm considers semantic meaning, and both may produce false positives when strings are syntactically similar but semantically different.

Table 5. Working the algorithms works on Martha, and Marhta strings.

Algorithm	Limitations
Simil	Computationally expensive for long strings
Jaro-Winkler	Less effective for long multi-token fields

VII. CONCLUSION AND FUTURE WORK

Duplicate records remain a major challenge for database systems and data-driven applications. This paper reviewed and compared two effective string-based similarity algorithms: Simil and Jaro–Winkler. The results show that Simil is more suitable for long textual fields with reordered tokens, while Jaro–Winkler excels in detecting short-string spelling errors. Overall, using such similarity algorithms allows database systems to automatically detect and reduce duplicate entries, improving data quality, consistency, and storage efficiency.

In practice, the best duplicate detection systems combine multiple similarity metrics with machine learning and clustering techniques. Future research should evaluate these algorithms on large-scale datasets and integrate them with AI-based duplicate detection models.

Authors' Contributions

AUTHOR CONTRIBUTIONS: Conceptualization, M.N.S.; Methodology, M.N.S., and M.F.A.; Validation, M.F.A.; Writing Original Draft Preparation, M.N.S.; Writing Review & Editing, M.F.A.; Supervision, M.F.A.

Conflict of Interest

The authors declare that there is no conflict of interest.

REFERENCES

[1] A. E. Monge ," Matching Algorithms within a Duplicate Detection System " , Bulletin of the Technical Committee on Data Engineering ,Vol. 23 No. 4 IEEE Computer Society,December 2000.

[2] B. Khan, A.Rauf, S.Shah, S.Khusro, "Identification and Removal of Duplicated Records" Article in World Applied Sciences Journal , January 2011.

[3] I. Ilyankou, J. Krajnak, "Comparison of Jaro-Winkler and Ratcliff/Obershelp algorithms in spell check," IB Extended Essay Computer ScienceUWC Adriatic, May 2014.

[4] D. Bharambe, S.Jain, A.Jain ," A Survey: Detection of Duplicate Record",International Journal of Emerging Technology and Advanced Engineering Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 11, November 2012)

[5] A. Elmagarmid, P. Ipeirotis, V.Verykios,"Duplicate Record Detection: A Survey" , IEEE Transactions On Knowledge And Data Engineering, Vol. 19, No. 1, January 2007.

[6] J. Soyemi, J. Adegboye, "Database Record Duplicate Detection System using Simil Algorithm"Article in International Journal on Computer Science and Engineering, February 2018.

[7] A. Susanto, N. Muliadi, B. Nugroho, "Comparison of String Similarity Algorithm in post-processing OCR" Journal of Applied Intelligent System , Vol. 8 No. 1, February 2023.

[8] E. Rahm, H. Hai Do, "Data Cleaning: Problems and Current Approaches",Bulletin of the Technical Committee on Data Engineering ,Vol. 23 No. 4 IEEE Computer Society,December 2000.

[9] X. Chu, I. Ilyas, S. Krishnan, J. Wang, "Data Cleaning: Overview and Emerging Challenges", SIGMOD'16, June 26-July 01, 2016, San Francisco, CA, USA © 2016 ACM. ISBN 978-1-4503-3531-7/16/06.