

Development of a Strategy for Duplication Search Based on Multiple Hierarchically Organized Approaches

Cephas A. S. Barreto(✉), Arnaldo S. B. Junior, Leonardo D. G. Melo, Maycon D. R. Santos, Gabriele R. Carvalho, Itamir M. B. Filho, Jose G. R. Neto, Ramon S. Malaquias, Andre M. Gurgel, and Jean M. M. Lima

Digital Metropolis Institute (IMD), Federal University of Rio Grande do Norte (UFRN), Natal, Brazil
ascom@imd.ufrn.br
https://portal.imd.ufrn.br

Abstract. Data deduplication is essential for records in business systems, as duplicate entries can distort analyses, harm relationships, and generate unnecessary costs. As organizations increasingly rely on diverse and large-scale data sources, the challenge of identifying and consolidating duplicate records has grown, making efficient deduplication a critical factor for maintaining data quality and operational efficiency. This paper proposes a hierarchical data deduplication approach to address this industry-wide challenge. The proposed approach, referred to here as the engine, integrates specialized rules, similarity measures, and large language models (LLMs) in sequential layers to analyze data and identify duplications accurately and efficiently. By leveraging a structured combination of techniques, the approach overcomes the limitations of traditional methods, which often depend on isolated and less adaptable solutions. The engine balances performance and accuracy, handling both straightforward and complex cases of duplication. Experimental results using a customer-simulated database demonstrate that integrating these advanced techniques improves adaptability and precision compared to conventional methods. This strategy shows significant potential for large-scale data deduplication, offering a more robust solution to an industry-wide challenge.

Keywords: Data deduplication · Expert Rules · Text Similarity · Large Language Models

1 Introduction

Data deduplication eliminates duplicate records in a database. In corporate systems, customer-related data, for example, includes identification details and relationship history. In this context, identifying, processing, and eliminating duplicate records is essential for maintaining a consistent relationship between the company and its customers [31].

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2025
O. Gervasi et al. (Eds.): ICCSA 2025, LNCS 15649, pp. 368–385, 2025.
https://doi.org/10.1007/978-3-031-96997-3_23

As organizations accumulate large volumes of data from different sources [21], such as sales systems, management systems, process control systems, and credit granting systems, maintaining the consistency and quality of that data becomes critical [12]. Inconsistent entries—including duplicates, fraud, typos, and legacy records—directly impact operational efficiency, regulatory compliance, and especially customer satisfaction [41]. When duplicate records occur, whether due to entries in different systems, variations in data format, or integration errors, problems arise, such as distortions in analysis [29], difficulties in managing customer relationships [14], and high costs with redundant resources [16,30].

Data deduplication is, therefore, an essential component in customer information management [8], requiring accurate and efficient processes to identify and consolidate duplicate records. However, traditional deduplication methods often use isolated approaches: rule-based methods [18,26] or machine learning techniques [2,11]. While these approaches have specific advantages, they have limitations regarding flexibility and adaptability. Rule-based methods, for example, are fast [1] and easily interpretable but may fail to identify more complex duplicates [28]. Machine learning techniques, especially large language models, offer greater flexibility in identifying duplicates but have a high computational cost [27,37].

This work presents an innovative hierarchical approach to customer data deduplication, featuring a search engine for duplicate detection. The proposal seeks to overcome the limitations of traditional deduplication systems by combining different techniques in a structured way: specialized rules, similarity measures, and large language models. The hierarchical approach allows organizing these techniques into progressive layers of complexity, optimizing the use of computational resources without compromising accuracy. Specialized rules are initially applied to capture simple duplicates quickly and efficiently; then, more sophisticated similarity metrics are used to deal with intermediate cases, while the last layer, with LLM models, is reserved for more complex cases with high semantic demands.

The experimental results demonstrate that the hierarchical combination of techniques not only improves the accuracy of the deduplication process but can also reduce the execution time, especially on large volumes of data. The proposed structure can, therefore, adapt to both simple and complex duplication scenarios, maintaining the balance between performance and accuracy. The results highlight the potential of this multi-tiered approach for comprehensive and adaptive data deduplication in large-scale enterprise environments.

The remainder of this work is organized as follows. Section 2 introduces fundamental concepts that support the development of the proposed approach. Section 3 reviews the main related works, with emphasis on techniques applied to customer data deduplication. Section 4 describes the architecture and components of the proposed search and identification engine. Section 5 details the experimental methodology. Section 4 presents and discusses the experimental results. Finally, Sect. 7 concludes the study and outlines possible directions for future research.

2 Related Concepts

This section briefly presents some of the concepts involved in building the proposed duplicate data identification engine: expert rules, textual similarity metrics, and Large Language Models (LLMs).

2.1 Expert Rules

Expert rules are structured guidelines derived from expert knowledge, used for decision-making and classification in computer systems [3,33]. They rely on patterns identified through experience and theoretical insights [5,7] and function as "if-then" statements to ensure consistency in automated processes.

Common in expert systems [15,34], these rules replace human judgment in diagnostics, classification, and recommendations. Unlike machine learning, which extracts patterns from data, expert rules are explicitly programmed [45]. In hybrid AI systems, they serve as strict rules for critical decisions, complementing learning-based approaches [39,44].

2.2 Textual Similarity Metrics

Textual similarity metrics quantify the degree of similarity between texts [25]. Editing-based metrics, such as Jaro-Winkler and Levenshtein distance [4,22,43], measure the number of operations (insertions, deletions, substitutions) needed to transform one text into another. These methods effectively identify minor spelling variations, commonly used in record matching and duplicate detection [32].

Another approach involves cosine similarity, a mathematical metric used in text comparison [36,47]. With the rise of machine learning, embedded numerical representations of text enable more precise similarity analysis by positioning semantically related words closer in a vector space [42]. Unlike character-based metrics, embeddings capture meaning-based relationships, making them particularly useful in detecting duplicate content that has been rewritten or paraphrased [19,20].

2.3 Large Language Models

LLMs are AI models trained on vast textual data to process and generate human-like text [24,35]. Using deep learning architectures like Transformers, they analyze complex linguistic structures, recognizing grammar, context, and semantics [38]. Their ability to detect nuanced textual similarities allows them to identify duplicate content, even when reworded or conceptually altered.

While highly flexible compared to expert rules and similarity metrics, LLMs are computationally expensive. Their usage should be limited to cases where more efficient methods fail, ensuring a balance between accuracy and processing time.

3 Related Work

The growing volume of data has intensified the need for effective duplicate identification to maintain business consistency. This section reviews related work, focusing on expert rules, similarity measures, and LLMs.

Expert Rules: The approach in [10] improves deduplication by applying expert rules individually and combining scores through information fusion, enhancing accuracy while reducing manual adjustments. Similarly, [17] classifies expert rule techniques by error type, automation potential, and update model, discussing trends in large-scale data cleaning.

Similarity Metrics: In [9], a similarity-based deduplication approach for digital libraries achieved a 188% improvement over traditional methods. [13] introduces a SHA-based method that removes redundant data while preserving coherence and optimizing resources. [40] proposes a locality-sensitive hashing (LSH) technique that combines textual and semantic similarity for efficient entity resolution. Additionally, [6] presents a two-step selection strategy (T3S) that enhances deduplication by balancing subsets and eliminating redundant pairs.

LLMs: [46] employs BERT in a hierarchical computing model for plagiarism detection, demonstrating effectiveness in ambiguous cases. [23] introduces BoostER, a cost-effective entity resolution system leveraging LLMs for query verification, making high-quality deduplication accessible without complex training or high costs.

Unlike most approaches that focus on a single technique, the proposed engine integrates expert rules, similarity metrics, and LLMs hierarchically. This structure enhances both accuracy and execution efficiency, offering a more comprehensive solution to data deduplication.

4 Proposed Engine

This section presents the proposed approach, a modularized engine developed to identify duplicated records. The engine operates as a service and is structured to perform a series of comparisons based on a sequence of rules, each designed to catch specific duplications.

Figure 1 presents the conceptual architecture of the proposed engine. This architecture has the engine as its main component, designed as a service that external systems can use via API and a user interface. The engine was also conceptually modeled to use an external LLM service in client comparisons when this technology is necessary. Finally, all comparisons performed are stored in a database and available for access via API and the user interface.

The engine is composed of three main modules, also called layers: expert rules, textual similarity, and LLM. These modules make up a set of rules that operate sequentially. Furthermore, to reduce the computational cost, when a duplication between a pair of clients is identified, the engine prevents comparisons made with more complex modules and returns the aforementioned pair as a duplicate. The following parts of this section present the characteristics of each module, aspects related to its construction, and details about the data flow.

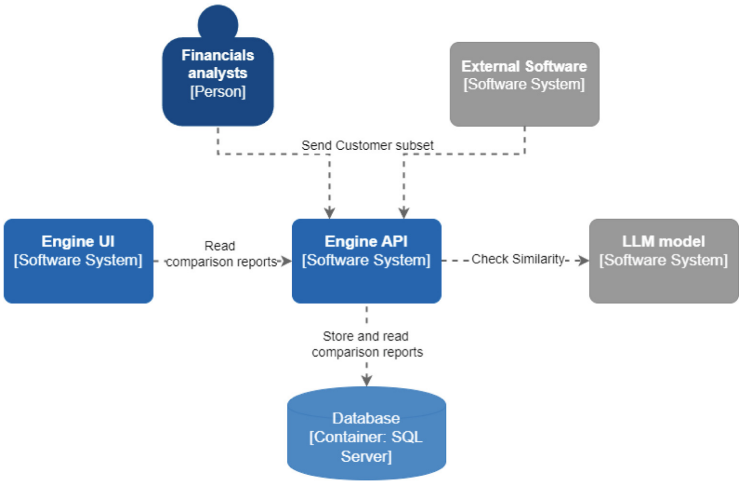

Fig. 1. Conceptual engine architecture.

4.1 Expert Rules Module

The first module of the engine aggregates a set of four expert rules, modeled to simulate cases of data duplication commonly found in a customer database. These rules simulate some well-known duplication cases in customer data and seek to capture duplications by equality, each linked to one or many data attributes. These rules were chosen and modeled based on superficial analyses of customer datasets available on the internet. The rules that make up the expert rules module are:

- **System id rule:** this rule represents the case in which an identifying attribute (id) from an old system needs to be maintained in the customer data after the system is modernized. Occasionally, two registrations made in the modern version of the system may refer to the same ID from the previous system. The rule, therefore, is triggered when the value in the attribute "IdOldSystem" from one client is identical to that from another client, signaling a possible case of duplication in the records
- **Bureau info rule:** this case represents the duplication of customers based on external credit identifiers (bureau). For this rule, two clients referencing the same bureau information are potentially duplicates. The rule is then applied when the registered bureau registration number ("bureau number") is identical between two clients, indicating a potential duplication;
- **Identification and contacts rule:** this case represents the duplication of identification data and contacts registered for a customer. This rule, therefore, is triggered when all attributes related to identification and contacts have identical data;
- **Identification and address rule:** this case represents the duplication of identification data and addresses registered regarding a customer. This

rule, therefore, is triggered when all attributes related to identification and addresses have identical data.

4.2 Textual Similarity Module

The second layer of the engine contains data comparison via similarity metrics. The technique chosen to compose this layer of the engine was the cosine metric, calculated on a text containing all the data from the two clients. This metric was chosen considering the results of preliminary experiments that investigated several approaches, including metrics based on the edition cited above.

As mentioned, the second layer of the engine uses cosine as a metric to check the similarity between customer data. To perform this calculation, each client has all its data transformed into a unique string separated by spaces. After this transformation, the resulting string embeddings are computed using a specific Autoencoder model specialized in short texts. After that, the distance cosine between the two sets of embeddings is calculated and normalized, returning a value between 0 and 1, indicating the similarity level between the two strings. This flow is detailed in Fig. 2.

4.3 Large Language Model Module

As mentioned, LLM models have a high semantic data analysis capacity and a high computational cost. The third layer of the engine uses this capability as a last resort to check for duplication between clients. At this layer, structured data from a customer pair is sent to the model. A well-defined prompt indicating broad comparison is used to ask the model to indicate whether the analyzed pair is potentially a duplicate. Additionally, a brief explanation of the result provided is requested.

It is important to mention that, despite being more computationally expensive, using an LLM model to compare more complex cases can bring benefits, as this type of technology cannot only return a result but also provide an explanation for the result. This ability can benefit the team responsible for evaluating effective duplication among the customers analyzed. Furthermore, it is possible to mitigate the difficulty over time by sending for analysis via LLM only in cases where the data is not clearly different.

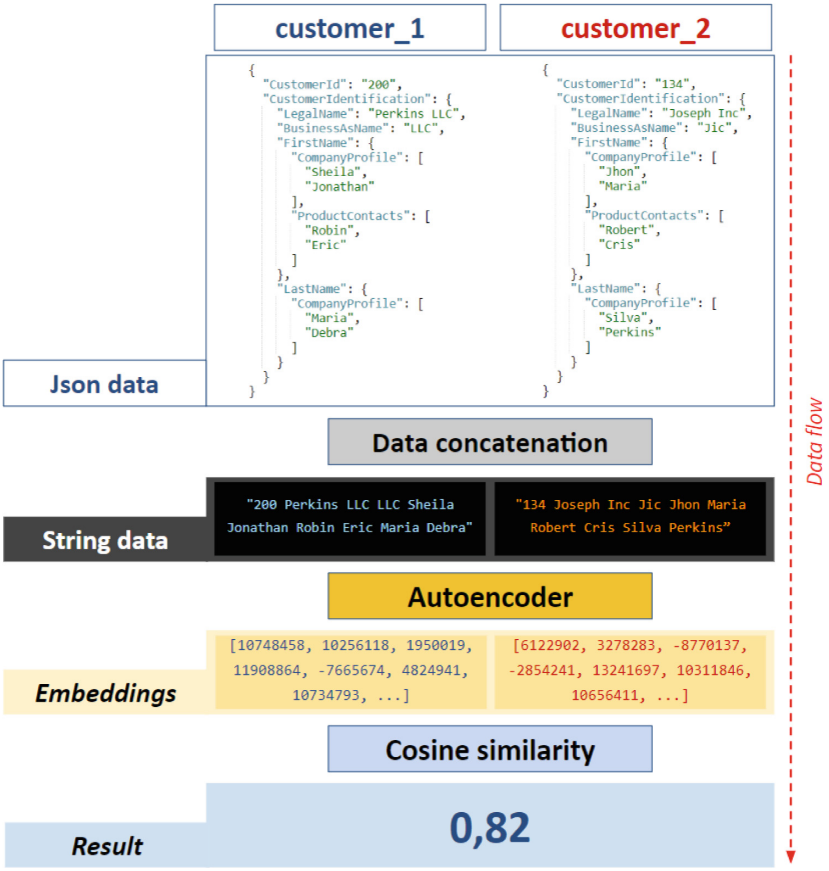

Fig. 2. Cosine similarity details.

4.4 Information Flow in the Engine

In an overview, the data flow of a pair of clients to be compared in the engine works according to the flow shown in Fig. 3. This flow and some of its characteristics will be detailed below.

First, data from the two received clients is sent to the first layer [1], which uses expert rules to identify duplications. The first layer rules are used in a well-defined sequence, being: [1.1] System id rule, [1.2] Bureau info rule, [1.3] Identification and contacts rule and [1.4] Identification and address rule. When a duplication is found in one of the rules, the flow of comparisons is stopped, and the identified rule is returned as a label for the possible duplication.

When the pair of compared clients is not captured by any rule in the first layer [1], their data is sent to the second layer [2], which uses the cosine metric as a way to check duplicity. In this layer, as already mentioned, data is transformed into texts, and their embeddings serve as a means for calculating the cosine,

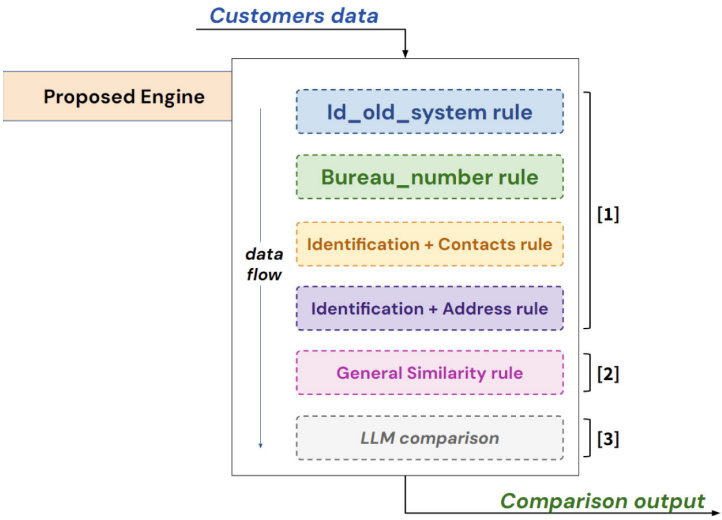

Fig. 3. Data flow inside the proposed engine.

which is returned in the form of a numerical value between 0 and 1. If the cosine value for the compared customer pair is greater than the established upper threshold, the flow of comparisons is interrupted, the referred pair is indicated as possibly duplicated, and the overall textual similarity is returned as a label for the duplicate found. Still, at this layer, when the cosine value of a client pair falls below the lower threshold, the comparison flow is also stopped, and the said client pair is labeled as non-duplicate.

Finally, if a pair of customers analyzed in the second layer has a cosine value below the upper threshold and above the lower threshold, its comparison is sent to the third layer [3], which uses an LLM model to evaluate whether said pair of customers are similar. The model analyzes the data based on a slightly open-ended prompt, and the result returned contains the overall assessment of duplicity (yes or no) and a brief explanation of this assessment.

4.5 Final Remarks About the Engine

The way the engine was modeled, based on a modular and sequential structure of different comparison techniques, allows for a series of gains compared to traditional ways of identifying duplicate data. Among the potential gains, we can mention:

- Mapping of specific needs and knowledge already existing in the business to identify duplicate data according to important attributes or sets of attributes;
- General syntactic and semantic analysis of data through comparison based on embeddings and cosine;

- Analysis of cases considered complex through the LLM model, including making it possible to write down the explanation of the decision regarding duplication;
- Reduction in the use of computational resources, since the most costly comparisons are only used when the previous layer indicates this way;
- General gain in performance in terms of accuracy as the engine combines capabilities from different technologies;
- Reduction of false positives, since the engine uses more open technologies (textual similarity and LLM) in a smaller number of cases;
- Reduction in computational time, given that the engine uses the most expensive layer (LLM) only in a restricted number of cases.

Based on the general context presented and the engine developed, we believe that the proposed solution can considerably improve the performance of duplicate identification systems, as it combines the performance capabilities of different approaches and manages to address time, one of the most significant difficulties in problems of duplication, in a balanced way.

5 Experimental Methodology

The implemented experimental methodology is designed to address the challenge of identifying duplicate clients in order to understand the benefits of using a hierarchical architecture that combines specialized rules, similarity measures, and large language models. The following sections present the experimental data, the comparative methods used in the analyses, and the configurations and metrics considered to analyze the results.

5.1 Experimental Data

The data used in the experiments were generated synthetically. The use of synthetic data in this study was motivated by two main factors: (1) the need to create controlled scenarios that allow for a precise and reproducible evaluation of the hierarchical model's capabilities, which would be difficult to achieve with real datasets; and (2) the constraints imposed by confidentiality and data protection requirements in corporate contexts.

Additionally, using synthetic data makes it possible to work with the texts in their original format, which would be difficult if a real dataset were used, as security and confidentiality measures often require the masking or encryption of personal information, making direct comparisons unfeasible between data such as name or telephone number, for example. The structure of synthetic generated clients are available on https://bit.ly/4iGSUfA.

As can be seen, customers have a structure segmented into five main parts: customer identification (CustomerIdentification), customer contacts (CustomerContacts), customer addresses (CustomerAddresses), customer Bureau numbers (BureauInfo), and the ID of the client in the old system (IdOldSystem). This

information can be internally segmented according to categories, which are: CompanyProfile, ProductContacts, or ProductBureau. These categories represent fictional data sections related to the customer's relationship with the company. Finally, this data formatting gave rise to the modeling of the expert rules mentioned in Sect. 4 - Proposal. These rules consider all data in each section and indicate equality between data when all information is exactly the same.

It is important to mention that all identifying attributes, such as: CustomerId, BureauInfo, and IdOldSystem, have a fixed number of characters. The number of characters in each fixed-length field is as follows:

- **CustomerId** : 3 characters;
- **BureauInfo**: 9 characters per BureauNumber;
- **IdOldSystem**: 12 characters;

5.2 Generated Datasets

Problems of searching for duplicate data have a quadratic nature (n^2), as they require comparing all customers with each other, for example. This particularity related to identifying duplicate data imposes a severe restriction on carrying out experiments since checking for duplicates in a dataset with n clients implies carrying out n x (n - 1) / 2 comparisons. For example, a dataset with just 10,000 customers would require 49,995,000 comparisons. If we hypothetically consider a 1 s execution time for comparing two customers with LLM, scanning all 49,995,000 necessary comparisons would take 578.65 d. This execution time would be unfeasible, even considering only 10 thousand customers.

The context presented indicated adopting a strategy to reduce the total number of comparisons. In this way, two sets of data were created: one called database, which simulates a company's customer database, and another called new customers, which simulates a group of more recently registered customers, that will be submitted to the investigation of duplicity. With this approach, all customers belonging to the new customer set must be compared with the database, which considerably limits the number of comparisons and, consequently, makes it possible to carry out experiments. As mentioned, two sets of data were created with the following characteristics:

- **Database**: set of 100 customer records, without internal duplications, which simulates a real, pre-existing database in a company;
- **New Customers**: set consisting of 100 customer records in which data that simulates duplication was purposely inserted according to specific scenarios.

The inserted duplications total 50 and represent a small fraction compared to the 10,000 comparisons necessary for a complete comparison of the two sets mentioned. Despite this, if we look at the quantities and scenarios described below, compared to the total of 200 existing customers in both sets, this is a considerable amount of duplication. Finally, the number of customers with duplications and the respective simulated duplication scenarios are:

- 5 duplicates via System id rule;
- 5 duplicates via Bureau info rule;
- 5 duplicates via Identification and contacts rule;
- 5 duplicates via Identification and address rule;
- 20 duplicates via similarity;
- 10 duplicates via LLM;

Finally, the synthetic data were carefully modeled to represent a context with a relatively large number of attributes. In addition, the inserted duplications seek to represent several simple and more complex contexts. In the latter case, simple analyses hardly capture the duplications, which justifies using LLM as the last layer for comparison.

5.3 Comparative Analysis

The comparative analysis was organized to evaluate the engine's performance in different configurations, considering the different processing layers. Thus, it is possible to verify whether the use of different comparison approaches jointly offers better performance in terms of accuracy and execution time when compared to the use of the same approaches in isolation.

To carry out the outlined investigation, each engine layer is evaluated individually on all data (100×100 comparisons). Furthermore, the engine with all its components working together is also tested, which totals the following versions to be tried:

- **Eng_EXP**: in this configuration, only the expert rules layer is applied;
- **Eng_SIM**: just the similarity layer;
- **Eng_LLM**: only the analysis layer with LLM;
- **Engine_COMPLETE**: complete engine.

The engine's performance was evaluated considering accuracy, represented by the total number of hits compared to the expected duplications. Furthermore, the number of false positives, that is, the number of false duplications identified, is also used as a criterion for analyzing the results. One last criterion analyzed was execution time, which indicates how viable each approach is regarding time to return results.

5.4 Environment and Technologies

Data was analyzed under the four aforementioned comparison settings, allowing for a comprehensive comparison of the efficiency and effectiveness of each approach. The development and execution environment for the experiments was based on Python, using libraries such as NumPy and SciPy for manipulating arrays and calculating cosine. Regarding the thresholds considered for the cosine, the following values were considered, based on preliminary experiments: upper threshold 0.7 and lower threshold 0.5.

Machine learning frameworks like Hugging Face Transformers have also been used to implement LLMs. The Autoencoder model used was paraphrase-MiniLM-L3-v2, and the LLM model used was Llama 3–13B, available on a local server running Ollama Server. The model temperature was set to 0.5, and the prompt and assistant are available on: https://bit.ly/4iGSUfA. The LLM module was developed using Python and based on the Ollama Server API. In this way, the data was sent to the server and compared using the configurations already presented.

The experiments were conducted on a desktop computer with the following technical specifications: Intel Core i5 processor, 16 GB RAM, and NVIDIA graphics card with 2 GB of memory. This standardization allows not only the analysis of accuracy and performance in terms of time but also the understanding of the relative effectiveness of each layer of the engine in identifying duplicate customers. In corporate environments, the availability of more powerful hardware would considerably reduce execution time. Despite this, it is important to verify the feasibility of experiments carried out even in an environment based on commercial hardware.

6 Results

This section presents the results observed in the experiments carried out. As mentioned, two sets were generated, each with 100 customers, and they were compared to each other, which generates a total of 10,000 comparisons. Another important aspect is related to comparative analysis, which considers the accuracy and execution time of each layer of the engine being executed in isolation as well as the complete engine. For simplicity, each engine layer and the entire engine will be called engine versions or simply versions.

6.1 Accuracy - Overall Performance

Table 1 presents the general result obtained by each engine version regarding successes and errors in positive cases (duplications) and negative cases (non-duplications). In that table, the first column on the left presents the identification of each case, and the second to fifth columns present each of the analyzed versions, including the versions with isolated layers (Eng_EXP, Eng_SIM and Eng_LLM) and also the version with the complete engine (Eng_COMPLETE).

Table 1. Results - Total comparisons

	Eng_EXP	**Eng_SIM**	**Eng_LLM**	**Eng_COMPLETE**
True Positives	20	23	40	48
False Positives	0	9	6	2
True Negatives	9950	9941	9944	9948
False Negatives	30	27	10	2

It is important to mention that the table refers to the total number of comparisons (10000), making no distinction between the specified duplication scenarios. Furthermore, the table's organization simply demonstrates the errors, denoted as false positives (non-existent duplications reported as true) and false negatives (duplications no longer identified). These errors are important to evaluate whether the performance of each version is consistent with the needs of the problem since identification errors are critical in the context of data duplication.

Regarding the results presented in Table 1, it is possible to observe that the version with all layers and located in the rightmost column, Eng_COMPLETE, obtained the best result. This version achieved 48 hits for duplications (positive) and 9948 hits for comparisons without duplication (negative), totaling 9996 hits among the 10,000 comparisons. Among the versions with just one layer, the Eng_LLM version stands out, which uses the LLM model to carry out comparisons. This version obtained 9984 hits, being very close to the overall performance of the complete engine. The two versions mentioned also had a low number of errors. While the Eng_COMPLETE version had only 4 errors (2 false positives and two false negatives), the Eng_LLM version had 16 errors, 6 false positives and 10 false negatives. This panorama demonstrates the superiority of the proposed version in terms of accuracy.

Regarding the Eng_EXP versions, which include expert rules, and Eng_SIM, which uses only the cosine similarity layer, it is possible to notice that the version with expert rules achieved better performance, reaching 9970 hits, 20 of which were true positives and 9950 true negatives. The Eng_sim version achieved a similar hit performance, with 9964 hits, of which 23 were true positives, and 9941 were true negatives. Regarding errors, it is noted that the Eng_Exp version had fewer errors, with 30 in total, while the Eng_SIM version had 36 errors in total. This performance demonstrates that, despite being similar in terms of hits, expert rules are less susceptible to errors, as they strictly limit duplication, while the version based on cosine similarity performs syntactic and semantic analysis during the comparison, which can harm the performance in terms of accuracy.

6.2 Accuracy - Performance by Duplication Scenarios

Table 2 presents the details of the results obtained by each version but from a more focused perspective on duplication scenarios. As already mentioned, scenarios were created that are strongly linked to each layer of the engine. The general structure of the table is similar to the previous table. However, the leftmost column details the duplication scenarios and the number of existing cases (in parentheses) per scenario. The other columns detail the successes of each version by duplication scenario. Finally, the last line of the table presents the total number of hits for each version and the percentage of these hits compared to the 50 cases of duplication entered.

Observing Table 2, it is possible to notice that the performance of the versions improves considerably as their complexity increases. The Eng_EXP version, for example, can only achieve 20 hits, which is the total number of duplication cases inserted in the duplication scenario, referring to the expert rules. Despite this,

Table 2. Results - Hits per layer and duplication scenarios

Duplication scenarios	Eng_EXP	Eng_SIM	Eng_LLM	Eng_COMPLETE
Expert rules (20)	20	0	20	20
General similarity (20)	0	19	11	19
Complex similarity (10)	0	4	9	9
Total (50)	20 (40%)	23 (46%)	40 (80%)	48 (96%)

this version does not have the flexibility to capture any other case of duplication, resulting in a performance of 40% accuracy compared to the 50 existing duplications. The Eng-SIM version, in turn, achieved a performance of 23 hits in total (46%), ranking third among the versions analyzed.

Still, in Table 2, it is noted that the Eng_LLM version achieved superior performance compared to the previously mentioned versions. This version achieved 40 hits, reaching 80% of existing cases. Finally, the version that uses all engine layers, Eng_COMPLETE, obtained the best performance among all the versions analyzed, achieving 48 hits in 50 possible cases, reaching 96% accuracy.

6.3 Runtime - Overall Performance

The execution time analysis was done based on the total time to perform all 10000 comparisons with each engine version. Table 3 presents the results in terms of execution time. The columns show the performance of each version, and the rows detail the average performance and total engine execution time, both in seconds.

Table 3. Results - Runtime overall performance

Runtime - seconds	Eng_EXP	Eng_SIM	Eng_LLM	Eng_COMPLETE
Runtime by comparison (avg)	0.003	0.235	25.200	2.977
Total runtime (10k comparisons)	28.33	2352.00	252000.00	29768.03

According to the data presented in the table, it can be seen that the fastest version among all those analyzed is Eng_EXP, composed only of the layer with expert rules, which has an average execution time of 0.003 s and a total time of 28.33 s to perform all 10000 comparisons. In second place is the Eng_SIM version, which had an average time of 0.235 s and a total time of 2352 s. In third place is the version with all engine layers, Eng_COMPLETE, with an average time of 2.97 s and a total time of 29768.03 s. Finally, as the most costly performance, the version that uses only LLM, with an average time of 25.20 s and a total time of 252000 s.

The results obtained demonstrate the simpler versions' superiority in terms of execution time. It is possible to notice a considerable increase in the total time, which goes from 28.33 s in the fastest version to 252000 s, or 70 h, in the most expensive version. The results presented in terms of time also indicate the impossibility of using strategies based solely on LLM, due to its high computational complexity. This aspect is crucial as it demonstrates the feasibility of using each approach and the need to articulate strategies that help reduce the total execution time.

It is important to highlight that the full version, Eng_COMPLETE, had a better performance in terms of time than the version that uses only LLM, Eng_LLM, as the strategy of filtering cases that go to the most costly layer (LLM) through cosine was used. As mentioned, when a comparison reaches the similarity layer, the value returned for the pair indicates whether it is a possible duplication (equal to or above the upper threshold, set at 0.7) or whether it should be denoted as non-duplicate (below the threshold lower, set to 0.5). This strategy allowed the full version to interrupt the execution flow in 8874 comparisons with similarity already observed as very low, reducing the engine's execution time.

6.4 Brief Discussion About the Results

The results in Tables 1 and 2 confirm the proposed engine's superior performance over all competing versions. This is evident in both overall success rates and error analysis across duplication scenarios. Notably, more complex isolated approaches, like similarity and LLM, had higher false positives and negatives, likely due to their semantic-based comparisons. In contrast, strict rule-based methods, which had no such errors, rely on easily identifiable patterns. This suggests a hybrid approach—applying simple rules to straightforward cases and complex rules to intricate cases—could enhance performance.

Regarding execution time, a viable approach must balance speed and accuracy. An engine that is fast but inaccurate is as ineffective as one that is precise but impractically slow. The proposed engine, running 8.5 times faster than the LLM-only version, outperforms isolated strategies while maintaining a significantly lower execution time than the most expensive alternative.

7 Conclusions and Future Work

This study introduces a hierarchical data deduplication approach that integrates expert rules, similarity metrics, and LLMs. By progressively structuring these techniques, the engine optimizes computational resources while ensuring high accuracy.

Experiments confirm the effectiveness of this architecture, with the full engine surpassing isolated approaches in both accuracy and efficiency. LLMs serve as a last resort for complex cases, while simpler layers handle straightforward duplications, minimizing false positives and negatives. Beyond technical strengths,

the engine's modular design enables adaptation to business needs, allowing rule customization and selective layer application based on dataset characteristics.

Future work will focus on validating the engine on real-world datasets to assess its practical applicability, developing adaptive algorithms for dynamic rule selection based on data properties, and enhancing the interpretability of LLM outputs to help users understand deduplication decisions. This hierarchical approach marks a significant advancement in scalable and adaptable data deduplication.

References

1. Abboura, A., Sahrl, S., Ouziri, M., Benbernou, S.: CrowdMD: crowdsourcing-based approach for deduplication. In: 2015 IEEE International Conference on Big Data (Big Data), pp. 2621–2627. IEEE (2015)
2. Akbar, M., Ahmad, I., Mirza, M., Ali, M., Barmavatu, P.: Enhanced authentication for de-duplication of big data on cloud storage system using machine learning approach. Clust. Comput. **27**(3), 3683–3702 (2024)
3. Amin, M.M., Stiawan, D., Ermatita, E., Budiarto, R.: Proposed threshold-based and rule-based approaches to detecting duplicates in bibliographic database. Bull. Electr. Eng. Inf. **13**(3), 2036–2047 (2024)
4. Bard, G.V.: Spelling-error tolerant, order-independent pass-phrases via the damerau-levenshtein string-edit distance metric. Cryptology ePrint Archive (2006). https://eprint.iacr.org/2006/204
5. Bhattacharya, I., Getoor, L.: Deduplication and group detection using links. In: KDD Workshop on Link Analysis and Group Detection (2004)
6. Bianco, G.D., Galante, R., Gonçalves, M.A., Canuto, S., Heuser, C.A.: A practical and effective sampling selection strategy for large scale deduplication. IEEE Trans. Knowl. Data Eng. **27**(9), 2305–2319 (2015). https://doi.org/10.1109/TKDE.2015.2416734
7. Blakely, T., Salmond, C.: Probabilistic record linkage and a method to calculate the positive predictive value. Int. J. Epidemiol. **31**(6), 1246–1252 (2002)
8. Boiński, P., Sienkiewicz, M., Bębel, B., Wrembel, R., Gałęzowski, D., Graniszewski, W.: On customer data deduplication: Lessons learned from a R&D project in the financial sector. In: CEUR Workshop Proceedings, vol. 3135 (2022)
9. Borges, E.N., de Carvalho, M.G., Galante, R., Gonçalves, M.A., Laender, A.H.: An unsupervised heuristic-based approach for bibliographic metadata deduplication. Inf. Process. Manage. **47**(5), 706–718 (2011)
10. Dinerstein, J., Dinerstein, S., Egbert, P.K., Clyde, S.W.: Learning-based fusion for data deduplication. In: 2008 Seventh International Conference on Machine Learning and Applications pp. 66–71. IEEE (2008)
11. Elouataoui, W., El Alaoui, I., El Mendili, S., Gahi, Y.: An end-to-end big data deduplication framework based on online continuous learning. Int. J. Adv. Comput. Sci. Appl. **13**(9), 281–291 (2022)
12. Even, A., Shankaranarayanan, G.: Dual assessment of data quality in customer databases. J. Data Inf. Qual. (JDIQ) **1**(3), 1–29 (2009)
13. Feng, D.: Overview of data deduplication. In: Data Deduplication for High Performance Storage System, pp. 9–23. Springer (2022). https://doi.org/10.1007/978-981-19-0112-6_2

14. González-Serrano, L., Talón-Ballestero, P., Muñoz-Romero, S., Soguero-Ruiz, C., Rojo-Álvarez, J.L.: Entropic statistical description of big data quality in hotel customer relationship management. Entropy **21**(4), 419 (2019)
15. Hair, K., Bahor, Z., Macleod, M., Liao, J., Sena, E.S.: The automated systematic search deduplicator (ASYSD): a rapid, open-source, interoperable tool to remove duplicate citations in biomedical systematic reviews. BMC Biol. **21**(1), 189 (2023)
16. He, Q., Li, Z., Zhang, X.: Data deduplication techniques. In: 2010 International Conference on Future Information Technology and Management Engineering, vol. 1, pp. 430–433. IEEE (2010)
17. Ilyas, I.F., Chu, X., et al.: Trends in cleaning relational data: consistency and deduplication. Found. Trends® Databases **5**(4), 281–393 (2015)
18. Jiang, Y., Lin, C., Meng, W., Yu, C., Cohen, A.M., Smalheiser, N.R.: Rule-based deduplication of article records from bibliographic databases. Database **2014**, bat086 (2014)
19. Kenter, T., De Rijke, M.: Short text similarity with word embeddings. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 1411–1420 (2015)
20. Kumar, A., Madanu, M., Prakash, H., Jonnavithula, L., Aravilli, S.R.: Advaita: bug duplicity detection system. arXiv preprint arXiv:2001.10376 (2020)
21. Kunz, W., et al.: Customer engagement in a big data world. J. Serv. Mark. **31**(2), 161–171 (2017)
22. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. In: Proceedings of the Soviet Physics Doklady, vol. 10, pp. 707–710 (1966)
23. Li, H., Li, S., Hao, F., Zhang, C.J., Song, Y., Chen, L.: Booster: leveraging large language models for enhancing entity resolution. In: Companion Proceedings of the ACM on Web Conference 2024, pp. 1043–1046 (2024)
24. Mahowald, K., Ivanova, A.A., Blank, I.A., Kanwisher, N., Tenenbaum, J.B., Fedorenko, E.: Dissociating language and thought in large language models. Trends in Cognitive Sciences (2024)
25. Majumder, G., Pakray, P., Gelbukh, A., Pinto, D.: Semantic textual similarity methods, tools, and applications: a survey. Computación y Sistemas **20**(4), 647–665 (2016)
26. Paganelli, M., Sottovia, P., Guerra, F., Velegrakis, Y.: Tuner: Fine tuning of rule-based entity matchers. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 2945–2948 (2019)
27. Rae, J.W., et al.: Scaling language models: methods, analysis & insights from training gopher. arXiv preprint arXiv:2112.11446 (2021)
28. Ravikanth, M., Korra, S., Mamidisetti, G., Goutham, M., Bhaskar, T.: An efficient learning based approach for automatic record deduplication with benchmark datasets. Sci. Rep. **14**(1), 16254 (2024)
29. Rehman, M., Esichaikul, V.: Duplicate record detection for database cleansing. In: 2009 Second International Conference on Machine Vision, pp. 333–338. IEEE (2009)
30. Reid, A., Catterall, M.: Invisible data quality issues in a CRM implementation. J. Database Mark. Custom. Strat. Manage. **12**, 305–314 (2005)
31. Reimer, K., Becker, J.U.: What customer information should companies use for customer relationship management? Practical insights from empirical research. Manag. Rev. Q. **65**(3), 149–182 (2015)
32. Rozinek, O., Mareš, J.: Fast and precise convolutional JARO and JARO-Winkler similarity. In: 2024 35th Conference of Open Innovations Association (FRUCT). IEEE (2024)

33. Shobha, K., Nickolas, S.: Integration and rule-based pre-processing of scientific publication records from multiple data sources. In: Satapathy, S.C., Bhateja, V., Mohanty, J.R., Udgata, S.K. (eds.) Smart Intelligent Computing and Applications. SIST, vol. 159, pp. 647–655. Springer, Singapore (2020). https://doi.org/10.1007/978-981-13-9282-5_61
34. Singh, P., Pal, N.R., Verma, S., Vyas, O.P.: Fuzzy rule-based approach for software fault prediction. IEEE Trans. Syst. Man Cybern. Syst. **47**(5), 826–837 (2016)
35. Su, C.Y., Bansal, A., Jain, V., Ghanavati, S., McMillan, C.: A language model of Java methods with train/test deduplication. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 2152–2156 (2023)
36. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining, 1st edn. Pearson, Boston (2006)
37. Thoppilan, R., et al.: LAMDA: language models for dialog applications. arXiv preprint arXiv:2201.08239 (2022)
38. Touvron, H., et al.: Llama 2: open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
39. Villena Román, J., Collada Pérez, S., Lana Serrano, S., González Cristóbal, J.C.: Hybrid approach combining machine learning and a rule-based expert system for text categorization. In: Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference. AAAI (2011)
40. Wang, Q., Cui, M., Liang, H.: Semantic-aware blocking for entity resolution. IEEE Trans. Knowl. Data Eng. **28**(1), 166–180 (2015)
41. Weis, M., Naumann, F., Jehle, U., Lufter, J., Schuster, H.: Industry-scale duplicate detection. Proc. VLDB Endow. **1**(2), 1253–1264 (2008)
42. Weston, J., Bengio, S., Usunier, N.: Large scale image annotation: learning to rank with joint word-image embeddings. Mach. Learn. **81**, 21–35 (2010)
43. Winkler, W.E., et al.: Overview of record linkage and current research directions. Bureau Census **25**(4), 603–623 (2006)
44. Wu, C.H.: Behavior-based spam detection using a hybrid method of rule-based techniques and neural networks. Expert Syst. Appl. **36**(3), 4321–4330 (2009)
45. Yakhni, S., Tekli, J., Mansour, E., Chbeir, R.: Using fuzzy reasoning to improve redundancy elimination for data deduplication in connected environments. Soft. Comput. **27**(17), 12387–12418 (2023)
46. Zhang, R., et al.: Hierarchical and pairwise document embedding for plagiarism detection. In: Yang, X., Wang, C.-D., Islam, M.S., Zhang, Z. (eds.) ADMA 2020. LNCS (LNAI), vol. 12447, pp. 148–156. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-65390-3_12
47. Zobel, J., Moffat, A.: Exploring the similarity space. In: Acm Sigir Forum. vol. 32, pp. 18–34. ACM New York, NY, USA (1998)