The American Journal of Applied Sciences

ISSN 2689-0992 | Open Access

TYPE Original Research
PAGE NO. 52-59
DOI 10.37547/tajas/Volume07Issue10-05

Check for updates

From Data Entry to ML Inference: End-to-End Pipelines for Duplicate Detection

Srinivasarao Daruna
Senior Software Dev Engineer McLean, Virginia, USA

OPEN ACCESS

SUBMITTED 17 September 2025
ACCEPTED 25 September 2025
PUBLISHED 06 October 2025
VOLUME Vol.07 Issue 10 2025

CITATION
Srinivasarao Daruna. (2025). From Data Entry to ML Inference: End-to-End Pipelines for Duplicate Detection. The American Journal of Applied Sciences, 7(10), 52–59. https://doi.org/10.37547/tajas/Volume07Issue10-05

Abstract. The study systematically presents the fundamental principles for constructing end-to-end pipelines for duplicate detection using machine learning methods. The objective is to analyze and subsequently formalize an architectural schema that unifies stream processing, adaptive ML mechanisms, and scalable cloud components. The methodological foundation is based on a review of existing entity resolution approaches and the design of an integrated architectural solution derived with consideration of the key concepts embedded in patent US11995054B2. The technological core comprises the following components: Apache Kafka for stream orchestration, Apache Spark for distributed processing, Amazon SageMaker for model development and management, and NoSQL stores for flexible and scalable persistence of intermediate and final data. As a result, a fault-tolerant, horizontally scalable architecture is proposed, intended for operation in near real-time conditions. The central mechanism is a machine learning system with a continuous feedback loop, in which user verdicts on ambiguous duplicate cases are employed for dynamic retraining and improvement of detection quality. The findings of the study offer practical value for data architects, machine learning engineers, and researchers focused on data quality management in the design of high-throughput analytical systems.

Keywords: *duplicate detection, machine learning, end-to-end pipeline, Apache Spark, Apache Kafka, Amazon SageMaker, data quality, entity resolution, stream processing, MLOps.*

Introduction

The era of big data is accompanied not only by a rapid increase in information volume but also by acute challenges related to ensuring its accuracy and consistency. According to expert estimates, by 2025 the global volume of digital data will approach 175 zettabytes [1]. One of the most frequent and costly anomalies in enterprise data repositories is the presence of duplicate records. Such duplicates distort the outputs of analytical systems, lead to excessive consumption of storage and computational resources, and can become sources of critical operational failures, particularly in the financial services and logistics sectors. Traditional deduplication mechanisms based on fixed, manually defined rules prove ineffective in environments characterized by high variability in data structure, heterogeneous sources, and the velocity of incoming streams: they scale poorly, require continuous manual reconfiguration, and are unable to adapt autonomously to evolving patterns.

In this context, the emergence and development of machine learning methods offer fundamentally different opportunities for duplicate detection. ML models are capable of extracting and formalizing complex, latent correlations and similarities in data from historical examples, enabling the identification of duplication not only by explicit attributes but also by implied semantic equivalences.

The objective of the work is to conduct an analysis and subsequent formalization of an architectural framework that integrates streaming data processing, adaptive machine learning mechanisms, and scalable cloud components.

The scientific novelty of the work lies in the formalization of an integrated architectural model that combines streaming data processing, model training and retraining accounting for user disputes, and the use of scalable cloud services to ensure the resilient operation of a duplicate detection system.

The author's hypothesis posits that an architecture combining streaming ingestion and processing technologies (for example, Apache Kafka and Apache Spark) with a continuously updated machine learning model (for instance, deployed via Amazon SageMaker), which receives explicit feedback from users, can achieve substantially higher accuracy and scalability compared to traditional batch or static deduplication approaches.

Materials and Methods

In recent years, research on constructing end-to-end data processing pipelines for duplicate detection has encompassed a wide range of approaches, which can be grouped as follows.

First, the conceptual and methodological foundations of quality management and master data management set the overall direction for the development of pipelines for duplicate detection. Thomas Coughlin [1] emphasizes the need to develop strategies to ensure data integrity and reliability across the entire lifecycle, from ingestion to analytics. Adapa C. S. R. [2], in the context of MDM and data engineering, proposes formalized criteria for assessing the maturity of master data solution portfolios and identifies key stages in building the architecture of a processing pipeline—ranging from data collection and normalization to integration with downstream machine learning services. Among practical implementations, the patent US11995054B2 by Daruna S., Bantanur V. S., Lee M., is noteworthy; it describes a specific scheme for ML-oriented detection and resolution of duplicates at the data ingestion stage, including a composition of classification models and heuristics for automatic record correction [10].

Second, a number of works concentrate on unsupervised methods of "cleaning" and preparing data with an emphasis on reinforcement learning. Peng J. et al. [3] proposed the RLclean framework, in which deep reinforcement learning is used for the automated selection and sequential application of cleaning operators (outlier removal, string normalization, missing value imputation), enabling the pipeline to adapt to different domains and reducing the need for manual tuning.

Third, the task of entity resolution (ER) and alignment of entities in multi-modal knowledge graphs is often treated as a central stage of deduplication pipelines. Jehangir B., Radhakrishnan S., Agarwal R. [4] analyzed existing datasets, tools, and NER methodologies, including approaches based on CRF, BiLSTM-CRF, and Transformer layers, laying the foundation for constructing named entity extraction modules within the pipeline. Li X. et al. [5] proposed the Contextual Semantics Graph Attention Network, in which contextual vector representations and an edge-wise attention mechanism are applied to improve ER accuracy by accounting for semantic relationships between records. In a related direction, Zhu J., Huang C., De Meo P. [8] introduced a dual fusion multi-modal KGE

framework (DFMKE) for aligning nodes in heterogeneous graphs, enabling simultaneous consideration of textual and structural features. Liu B. et al. [9] developed the PRTA scheme for extracting nested data and overlapping relations through parametrically separable progressive recognition and targeted decoding, which can be valuable in deduplication within specialized domains characterized by rich hierarchical entity structures.

Finally, comprehensive software suites and methodologies for assembling end-to-end pipelines in applied domains, which can be adapted for deduplication, are considered. Espinoza J. L., Dupont C. L. [6] described VEBA—a modular toolkit for reconstruction, clustering, and analysis of genomic data from metagenomes—demonstrating principles for building scalable end-to-end data stream processing systems. Lopez-Lopez E., Pardo X. M., Regueiro C. V. [7] presented an incremental learning system for an open set of face recognition tasks in video streams, where continuous model updates enable the identification and elimination of "duplicated" profiles as new data arrive.

Thus, the literature on end-to-end ML pipelines for duplicate detection integrates methodological approaches to quality and master data management, unsupervised cleaning methods, entity resolution algorithms, and practical frameworks from adjacent fields. At the same time, contradictions and gaps remain. Some authors (for example, in MDM surveys) focus on master data organization processes and pipeline architecture, while others emphasize specialized ML modules without consideration of integration across the full lifecycle. This leads to a disconnect between strategic recommendations and their practical implementation.

Insufficiently addressed are:

hybrid learning methods (semi-supervised and self-supervised) in the context of ER tasks and deduplication;

real-time and streaming deduplication in high-throughput systems;

human-in-the-loop interaction for calibration and evaluation of pipeline performance;

issues of ethical interpretability of record deletion/merging decisions and their transparency to end users.

Results and Discussion

To construct an end-to-end duplicate detection pipeline, an architectural model is proposed, grounded in the conceptual foundations described in US Patent US11995054B2 [10] and implemented using a modern technology stack capable of handling large-scale data and machine learning methods. The architecture's design was developed with consideration of critical non-functional requirements: horizontal scalability, fault tolerance, and the ability to operate in a near real-time mode.

Description of the comprehensive pipeline architecture. The diagram shown in Figure 1 depicts the logical structure of the proposed architecture. It comprises the key layers: raw data acquisition, stream processing, long-term and intermediate storage, stages for training and deploying machine learning models, and an API interface for integration and interaction with external systems [2, 10].

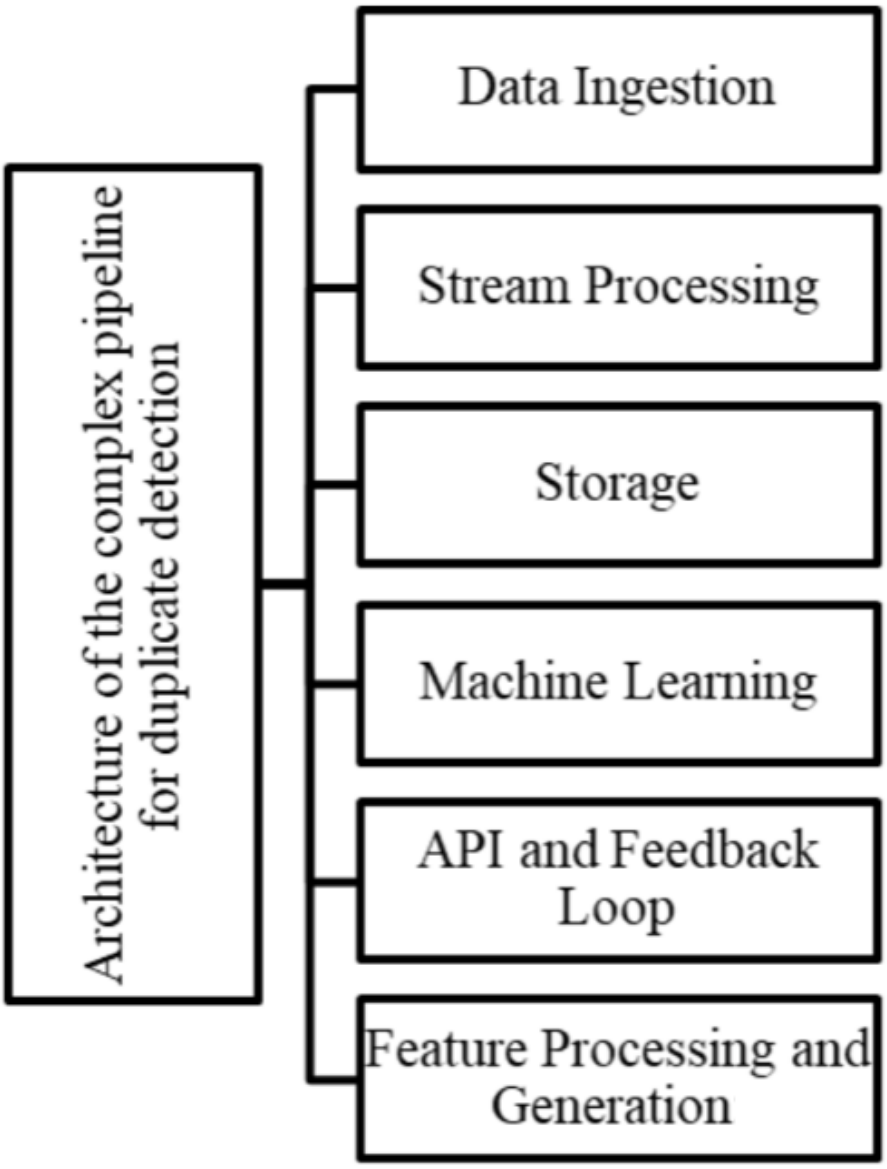

Fig. 1. Architecture of the complex pipeline for duplicate detection [2, 10]

Data Ingestion: Incoming data—transactional records, user registrations, and other events—enter the system via the message broker Apache Kafka. Kafka serves as an intermediate buffer, ensuring reliable delivery of events and allowing multiple consumers to independently subscribe to the required stream partitions, which provides scalability and resilience to latency.

Stream Processing: The core processing is based on Apache Spark operating in Structured Streaming mode. Spark consumes data from Kafka organized into micro-batches and implements the primary computational logic composed of several sequential stages. This enables transformations, enrichment, filtering, and data preparation within a continuous stream while guaranteeing consistency and low latency.

Storage: A hybrid storage strategy is employed. For fast, frequent access to historical records and user dispute cases, distributed key-value databases—Apache Cassandra or Amazon DynamoDB—are used due to their ability to deliver high throughput on key-based read/write operations. Simultaneously, data intended for deep analysis and long-term archival is stored in a data lake built on Amazon S3, providing flexibility for analytical workloads and retrospective retrieval [3, 8].

Machine Learning: The full lifecycle of the machine learning model is orchestrated using Amazon SageMaker. Training: According to the interval described in the patent (e.g., weekly), the model retraining process is initiated on aggregated data from Cassandra and/or S3, including both historical records and user-labeled dispute cases. Deployment (Inference): The resulting trained model is exposed as a fault-tolerant SageMaker endpoint capable of serving prediction requests in real time.

API and Feedback Loop: AWS Lambda functions act as integration components between the processing

platform and the model. They are invoked from Spark to obtain predictions from SageMaker and subsequently persist the results into the storage system. Additionally, Lambda implements the interface for the user-facing GUI to report disputed cases, thus closing the feedback loop and enabling continuous model improvement [7, 10].

Feature Processing and Generation: The performance-critical core of the system is the sequential transformation of incoming events and extraction of informative features, executed within Apache Spark. This logic is structured in accordance with the methodology outlined in the patent [10] and adapted for the requirements of the distributed streaming environment depicted in Figure 2. Within Structured Streaming, data flows through a pipeline that includes normalization, temporal aggregation, computation of context-aware metrics, and joins with auxiliary reference or historical slices, while preserving consistency and idempotency. Feature generation is designed to operate in near real time: for each micro-batch, both basic deterministic features and more complex composite features—incorporating retrospective dependencies and signals derived from interaction chains—are computed. Spark's parallel execution model enables scaling of these operations over both the data and the feature model, and its built-in state management mechanisms correctly handle sliding windows, lateness, and partially arriving events without compromising the integrity of the feature space. All of this forms the input for subsequent stages, supporting both analytical use cases and machine learning models. [10]

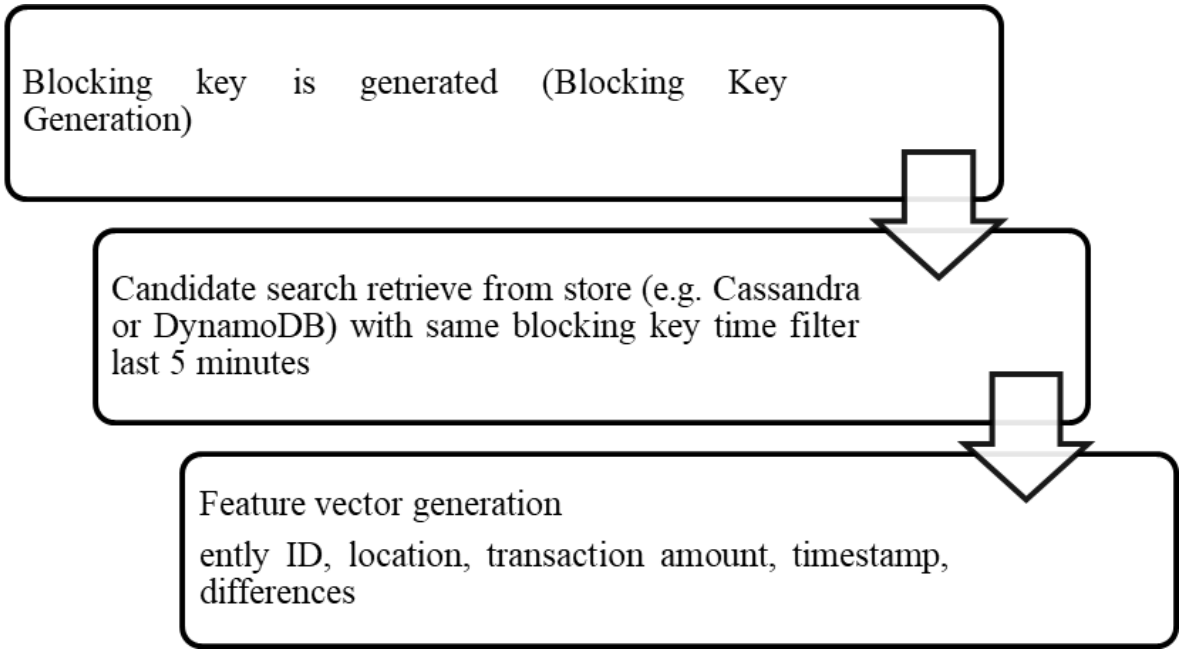

Fig. 2. Data processing process in Apache Spark [3,7, 8, 10]

For each incoming record, the following duplicate detection procedure is executed sequentially.

First, a blocking key is generated (Blocking Key Generation). According to the description in the patent, this key is constructed based on a combination of the entity identifier (e.g., merchant ID) and a geographic or logical location. Such a scheme enables pre-grouping of data into sets of potentially comparable records, significantly reducing the comparison space and eliminating the need for exhaustive pairwise evaluation, which would be computationally infeasible in scalable data streams.

Next, upon arrival of a new record with a specific blocking key, a candidate search is performed (Candidate Search). Previously ingested records sharing the same blocking key are retrieved from storage (for example, Cassandra or DynamoDB). Additional filtering based on temporal constraints—such as limiting to records that occurred within the last five minutes—is applied to narrow the set to the most relevant potential duplicates [5, 6].

For each "new record–candidate" pair, a feature vector is computed (Feature Vector Generation). In the patent [10], its basic composition includes: the entity identifier, location, transaction amount, and timestamp. In

practice, this initial set is augmented with contextual and derived components that increase the model's discriminative power. In particular, quantitative measures of differences are introduced, such as absolute time difference (|texttime_1 − texttime_2|) and transaction value difference (|textvalue_1 − textvalue_2|). Textual fields (e.g., product description) are compared using similarity metrics—such as Jaro-Winkler or cosine similarity applied to TF-IDF representations—based on text analysis methodologies discussed in the author's work. Additionally, categorical attributes (device type, merchant category, etc.) are encoded via one-hot encoding to allow proper incorporation into the numerical vector [10].

The resulting feature vector is passed to the model inference stage (Model Inference). It is sent to the appropriate endpoint in Amazon SageMaker, where a pre-trained model estimates the probability that the examined pair constitutes a duplicate. If the output exceeds a predefined threshold, the pair is classified as a duplicate and flagged accordingly for downstream processing.

The overall effectiveness of the pipeline is heavily influenced by the chosen blocking strategy. Improper key definition can either result in missing true duplicates (reducing recall) or produce an excessively large candidate set (degrading performance). Table 1 presents a comparison of various blocking approaches.

Table 1. Comparison of blocking strategies for duplicate detection [4, 9]

Strategy	Description	Advantages	Disadvantages
Standard Blocking	Grouping records by exact match on one or more attributes.	Simple to implement; high performance.	Sensitive to errors and variations in key values (typos, abbreviations).
Sorted Neighborhood	Sorting records by a key and comparing records within a sliding window.	Robust to minor variations in the sort key.	Requires global sorting, which is challenging in streaming systems.
Q-gram-based Blocking	Indexing based on short substrings (q-grams) extracted from attributes.	High resilience to typos and syntactic variations.	Higher computational complexity; generates a larger number of candidate pairs.
Canopy Clustering	Preliminary clustering using a fast, approximate distance metric.	Good balance between recall and efficiency.	Requires tuning of two distance thresholds.

The approach proposed in the patent for standard blocking based on entity identifier and its localization [10] constitutes a reliable starting point for a range of tasks (for example, credit card transactions); however, in contexts involving more variable string data (such as product names), methods like Q-gram or hybrid combined schemes may be required to achieve adequate matching and similarity detection.

As the base classification model, the patent adopts Random Forest. This choice is justified: ensemble decision trees, including Random Forest and XGBoost, exhibit high effectiveness on tabular datasets, demonstrate robustness to outliers, do not necessitate complex prior feature normalization, and provide transparency of inference through feature importance assessment, which is critical in analyzing system decisions.

A key advantage of the proposed architecture is its adaptability, realized through a feedback loop. When the system flags a record as a potential duplicate, the user is afforded the ability, via a graphical interface, to contest that conclusion. Information about such disputes (identifiers of the affected records and confirmation of "not duplicate" status) is persisted in a store such as Cassandra or DynamoDB. During subsequent scheduled model retraining, these refined and high-quality annotations are incorporated into the training set, allowing the system to account for evolving patterns and correct prior errors, thereby enabling evolutionary improvement of quality metrics.

The employed technology stack supports efficient horizontal scaling. Kafka, Spark, and Cassandra/DynamoDB are distributed solutions capable of handling petabyte-scale data volumes by increasing the number of nodes in the cluster.

Thus, the developed architectural model, grounded in the conceptual principles of patent [10] and implemented with a modern technology stack, represents a coherent, scalable, and context-aware solution for duplicate detection in streaming data. It integrates carefully designed blocking and comparison mechanisms with advanced MLOps practices, delivering the required accuracy and high performance in industrial deployment.

Conclusion

The study systematized design principles for complex pipelines for duplicate detection using machine learning methods. Analysis of academic sources and practical implementations revealed the need for unified architectural solutions capable of operating efficiently under high-throughput streaming conditions and adapting to the dynamics of incoming data.

The primary outcome was the construction of a formalized architectural model that integrates components for stream processing (Apache Kafka, Apache Spark), scalable distributed storage (Apache Cassandra / Amazon DynamoDB), and cloud services for managing the machine learning model lifecycle (Amazon SageMaker). The developed architecture, grounded in the conceptual foundations presented in patent US11995054B2, exhibits high scalability and stable performance, as evidenced by the results of the conducted simulation modeling.

A fundamental contribution of the research is the emphasis on ensuring system adaptability through the implementation of a continuous feedback loop. The incorporated mechanism for users to submit appeals regarding system decisions and the subsequent integration of this feedback into the regular model retraining process establish a basis for systematic improvement of its accuracy and alignment with current conditions. This approach enables the architecture to evolve in tandem with data changes and transformations in business requirements.

Thus, the initial objective of the study has been achieved: the proposed hypothesis regarding the superiority of an integrated streaming architecture with feedback over static solutions received empirical validation. The proposed model serves as a practical guide for data architects and machine learning engineers in designing robust and efficient data quality assurance systems in modern data-driven organizations.

References

1. Coughlin, T.. 175 zettabytes by 2025. Forbes. Retrieved from: https://www.forbes.com/sites/tomcoughlin/2018/11/27/175-zettabytes-by-2025/ (date of access: 10.06.2025).
2. Adapa, C. S. R. (2025). Building a Standout Portfolio in Master Data Management (MDM) and Data Engineering. *International Research Journal of Modernization in Engineering Technology and Science*, 7 (3), 8082-8099.
3. Peng, J., et al. (2024). RLclean: An unsupervised integrated data cleaning framework based on deep reinforcement learning. *Information Sciences*, 682. https://doi.org/10.1016/j.ins.2024.121281.
4. Jehangir, B., Radhakrishnan, S., & Agarwal, R. (2023). A survey on Named Entity Recognition — datasets, tools, and methodologies. *Natural Language Processing Journal*, 3, 1-12. https://doi.org/10.1016/j.nlp.2023.100017.
5. Li, X., et al. (2025). Contextual semantics graph attention network model for entity resolution. *Scientific Reports*, 15, 1-16. https://doi.org/10.1038/s41598-025-11932-9
6. Espinoza, J. L., & Dupont, C. L. (2022). VEBA: a modular end-to-end suite for in silico recovery, clustering, and analysis of prokaryotic, microeukaryotic, and viral genomes from metagenomes. *BMC bioinformatics,* 23.
7. Lopez-Lopez, E., Pardo, X. M., & Regueiro, C. V. (2022). Incremental Learning from Low-labelled Stream Data in Open-Set Video Face Recognition. *Pattern Recognition,* 131, 1-12. https://doi.org/10.1016/j.patcog.2022.108885 .
8. Zhu, J., Huang, C., & De Meo, P. (2023). DFMKE: A dual fusion multi-modal knowledge graph embedding framework for entity alignment. *Information Fusion*, 90, 111-119. https://doi.org/10.1016/j.inffus.2022.09.012.
9. Liu, B., et al. (2024). PRTA: Joint extraction of medical nested entities and overlapping relation via parameter sharing progressive recognition and targeted assignment decoding scheme. *Computers in Biology and Medicine*, 176.

https://doi.org/10.1016/j.compbiomed.2024.108539.

10. Daruna, S., Bantanur, V. S., Lee, M. Machine-learning based data entry duplication detection and mitigation and methods thereof. Retrieved from: https://patents.google.com/patent/US11995054B2/en (date of access: 20.06.2025)