A Token-Based Framework for Detecting and Eliminating Duplicate Data in Big Data Analytics Using Machine Learning Techniques

Dr. J. Jebamalar Tamilselvi [1]
Dr. K. Sutha [2]
Dr. S. Sweetlin Susilabai [3]
Dr. Surya Susan Thomas [4]
Mr. Raman Raguraman [5]

[1,2,3] Department of Computer Science s/w Cyber Security, Faculty of Science and Humanities, SRM Institute of Science and Technology, Ramapuram, Chennai
[4] Assistant Professor, Patrician College of Arts and Science, Chennai
[5] Assistant Professor, Deputy Dean, Faculty of Engineering & Computer Technology, AIMST University, Malaysia

Abstract
Effective data management is essential in Big Data Analytics, where vast amounts of data are collected, processed, and analyzed to extract valuable insights. Duplicate data presents significant challenges, including increased storage costs, diminished analytical efficiency, and compromised data quality, all of which can lead to erroneous decision-making. Conventional deduplication techniques based on exact matching or simplistic heuristics often fall short in addressing data variability, semantic equivalence, and the massive scale typical of big data. This research proposes a token-based framework enhanced by a hybrid machine learning approach, integrating token-based strategies with both supervised and unsupervised learning techniques to accurately detect and eliminate duplicate data in Big Data Analytics environments.

Keywords: Token-based, Machine Learning, Big Data, Duplicate Data

Review of Literature

The rapid expansion of big data has exacerbated the problem of duplicate records, adversely affecting storage utilization, query performance, and overall data quality. As a result, detecting and removing duplicates has become a vital preprocessing step in data analytics pipelines.

Duplicate detection, also known as entity resolution, plays a foundational role in data cleaning and integration. Elmagarmid et al. (2007) established a structured methodology focusing on preprocessing, similarity measurement, and rule-based elimination. Although effective for traditional databases, these approaches lack the scalability required for the volume, velocity, and variety characteristic of modern big data systems. To address this, Kejariwal et al. (2012) extended duplicate detection into distributed computing environments using Hadoop and MapReduce, enabling parallelized tokenization, similarity calculations, and deduplication across billions of records.

Token-based similarity measures are widely adopted due to their ability to handle unstructured and semi-structured data. Metrics such as Jaccard similarity, Cosine similarity, and TF-IDF represent data as token sets or vectors, facilitating approximate fuzzy matching. Chaudhuri et al. (2003) demonstrated these methods' robustness and efficiency in online data cleaning. However, the computational burden of exhaustive pairwise comparisons in large datasets prompted the development of blocking and indexing techniques like sorted neighborhood and canopy clustering. Christen (2012) highlighted that these techniques significantly reduce the comparison space, making large-scale deduplication more tractable.

In recent years, machine learning has been increasingly leveraged to enhance duplicate detection performance. Supervised classifiers, including Support Vector Machines (SVMs), Random Forests, and Neural Networks, have been employed to categorize record pairs using engineered similarity features. Sarawagi and Bhamidipaty (2002) explored active learning for interactive deduplication, while unsupervised clustering algorithms such as DBSCAN and K-Means have effectively grouped similar records without labeled data, as noted by Bhattacharya and Getoor (2007). Current research trends favor hybrid frameworks combining rule-based token matching with adaptive machine learning models to boost accuracy and generalization capabilities.

The scalability of these approaches has been facilitated by big data processing platforms like Apache Hadoop and Spark, which support distributed and real-time deduplication workflows. Frameworks integrating sequential token-based methods, clustering, and blocking have demonstrated significant improvements in detection accuracy and data quality within data warehouse contexts. Nevertheless, challenges persist in optimizing precision and recall, minimizing false positives and negatives, and accommodating evolving or heterogeneous data sources. Dong et al. (2015) emphasize the need for scalable, domain-agnostic evaluation frameworks, while incremental deduplication and schema evolution continue to drive ongoing research. Integrating token-based techniques with machine learning within scalable architectures remains at the forefront of duplicate detection research in big data analytics.

Proposed Architecture Diagram

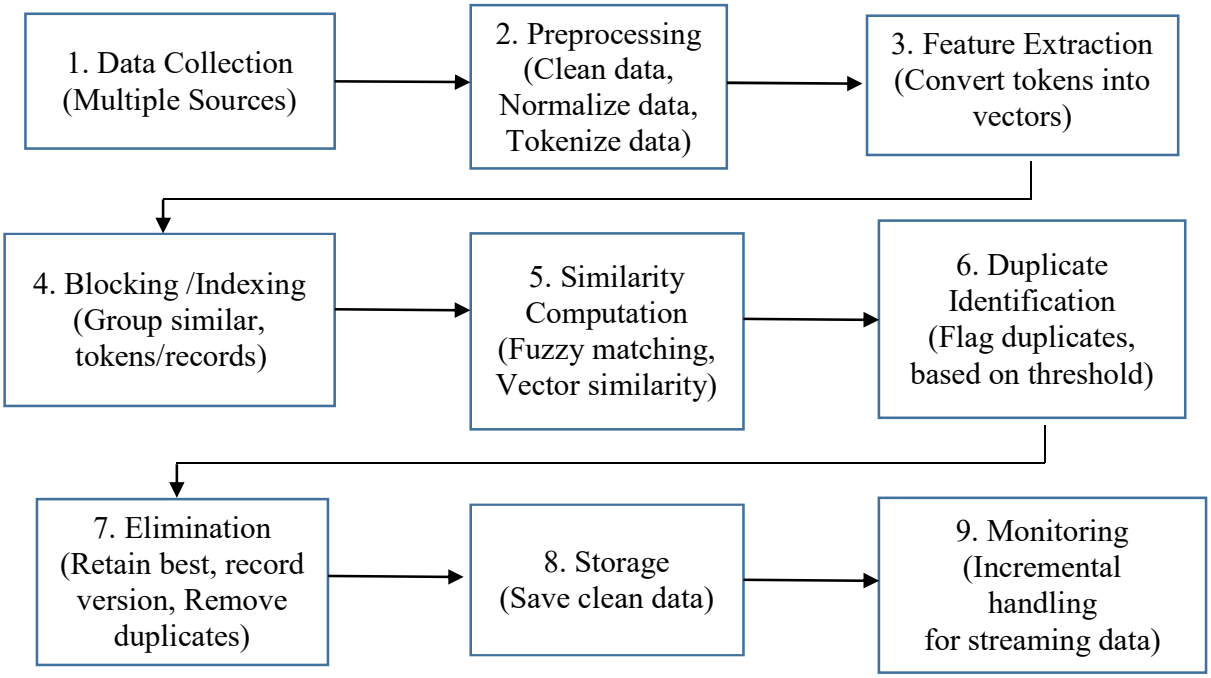
The novelty of this research lies in integrating tokenization mechanisms with machine learning models and clustering techniques, creating a highly adaptive and scalable solution for duplicate detection and elimination tailored to the challenges of Big Data Analytics.

1. Data Collection

The deduplication framework begins with data collection, gathering raw data from various heterogeneous sources such as relational databases, data lakes, web APIs, sensor networks, and unstructured repositories like documents and web pages. As highlighted by Elmagarmid et al. (2007), the diversity and scale of modern data require scalable ingestion techniques capable of handling structured, semi-structured, and unstructured formats with varying quality.

2. Preprocessing

Next, preprocessing prepares the raw data for accurate and efficient duplicate detection. This stage includes cleaning (addressing missing values, noise, and inconsistencies), normalization (standardizing formats such as dates, numbers, and text cases), and tokenization (splitting data into smaller units like words or n-grams). Effective cleaning minimizes error propagation (Elmagarmid et al., 2007), while tokenization enhances fuzzy matching and pattern recognition in noisy or semi-structured contexts (Chaudhuri et al., 2003). The importance of a staged preprocessing strategy to improve deduplication accuracy in large-scale settings is underscored by prior research (2008).



3. Feature Extraction

In the feature extraction phase, tokenized data are transformed into structured, high-dimensional feature vectors that capture syntactic and semantic characteristics for similarity assessment. Methods such as TF-IDF weight tokens based on frequency and distinctiveness, emphasizing discriminative terms. Semantic embeddings like Word2Vec and FastText (Mikolov et al., 2013) encode contextual relationships, enabling the detection of semantically equivalent near-duplicates despite surface-level differences. The 2009 study demonstrated that combining token-based feature extraction with clustering enhances the precision of duplicate grouping by improving record comparability and separability.

4. Blocking/Indexing

To reduce computational overhead, the framework employs blocking and indexing**,** which partitions data into smaller subsets, limiting similarity comparisons to within these blocks. Techniques like the Sorted Neighborhood method and Canopy Clustering (McCallum et al., 2000) create overlapping clusters using loose similarity thresholds to ensure potential duplicates are grouped together. Christen (2012) highlighted that blocking significantly decreases complexity while preserving recall, and integrating blocking with token-based clustering has proven effective for balancing scalability and accuracy in large datasets.

5. Similarity Computation

During similarity computation**,** various metrics quantitatively assess the resemblance between record pairs. Metrics such as Jaccard Similarity (token overlap), Cosine Similarity (vector angle in TF-IDF or embeddings), and Edit Distance (Levenshtein Distance for character-level comparison) are used. The 2010 research integrated these metrics into a scalable framework, minimizing unnecessary comparisons while maintaining accuracy, thus effectively handling data heterogeneity and noise.

6. Duplicate Identification

Duplicate identification classifies record pairs based on similarity scores and thresholds that balance false positives and false negatives. Machine learning models improve this process by learning complex decision boundaries from labeled data. Techniques include probabilistic graphical models (Bhattacharya & Getoor, 2007), supervised classifiers like Random Forests, SVMs, Gradient Boosting Machines, and deep learning architectures such as Siamese networks that model similarity directly. These approaches enable scalable, adaptive duplicate detection suited for complex and evolving datasets in big data environments.

7. Elimination

In the elimination step, the framework selects a canonical record per duplicate group to maintain data quality and integrity. Selection criteria consider completeness, recency, and source reliability. Traditional rule-based heuristics (as discussed in the 2011 paper on handling duplicate data) rank records by richness and context sensitivity. More recent advancements employ machine learning ranking algorithms such as Gradient Boosted Decision Trees (GBDT) and RankNet, trained on annotated data to optimize selection policies. Reinforcement learning methods further enable dynamic adaptation based on feedback, moving beyond static rules to data-driven elimination strategies.

8. Storage

The cleaned, deduplicated data are then stored in scalable systems such as data warehouses (Amazon Redshift, Google BigQuery), NoSQL databases (MongoDB, Cassandra), and cloud storage platforms (AWS S3, Azure Blob Storage). Tools like Apache Hive facilitate SQL-like queries over distributed file systems (e.g., HDFS), supporting analytical workflows. From a

machine learning standpoint, integration with feature stores and metadata repositories ensures consistent reuse of cleaned data and model outputs, maintaining traceability and enabling explainable, trustworthy workflows (Dong & Srivastava, 2015).

9. Monitoring

Finally, the framework incorporates monitoring to maintain deduplication effectiveness in real-time or streaming environments. Platforms such as Apache Kafka, Flink, and Spark Streaming enable scalable incremental processing of incoming data, applying cleaning, tokenization, and similarity comparisons on-the-fly. Reinforcement learning and online learning algorithms dynamically adjust similarity thresholds and blocking strategies based on feedback, enhancing precision and recall even under evolving data distributions (concept drift). This machine learning-driven monitoring ensures robustness, scalability, and accuracy in ongoing data quality management (Kejariwal et al., 2012; Zhang et al., 2020).

The proposed framework leverages advanced tokenization and machine learning methods to significantly improve duplicate detection accuracy for both exact and near-duplicates. It supports scalable, distributed processing of massive datasets, reducing redundancy and optimizing analytics performance. Machine learning-driven adaptability enables models to handle diverse data structures and evolving duplication patterns, enhancing robustness across domains such as healthcare, finance, and retail. Seamless integration with modern big data technologies and real-time systems ensures practical applicability in dynamic analytics pipelines, delivering a comprehensive, high-accuracy, scalable, and adaptable deduplication solution for large-scale data environments.

Methodology Used in the Framework

The framework integrates a multi-step methodology combining traditional data processing with advanced machine learning (ML) algorithms to efficiently detect and eliminate duplicates in large-scale, heterogeneous datasets.

1. **Data Collection and Integration**
   Data is gathered from multiple sources such as databases, APIs, and unstructured documents. Scalable pipelines ensure data consolidation but ML algorithms are not directly applied at this stage.
2. **Preprocessing and Tokenization**
   Data is cleaned and standardized, and tokenized into units (words, n-grams). While this step is largely rule-based, preprocessing improves ML model input quality downstream.
3. **Feature Extraction**
   - TF-IDF Vectorization: Converts text into weighted vectors to highlight important tokens.
   - Word Embeddings (Word2Vec, FastText): Capture semantic relations between tokens by embedding them into continuous vector spaces.
     These features provide rich numerical representations used as input for ML models.

4. **Blocking and Indexing**
   Efficient blocking methods like Canopy Clustering and Sorted Neighborhood reduce comparisons. Though primarily algorithmic, some frameworks use unsupervised clustering algorithms (e.g., K-Means) to group similar records.
5. **Similarity Computation**
   Calculates similarity scores using metrics like Jaccard, Cosine Similarity (on embeddings), and Edit Distance. These scores form feature inputs for ML classifiers.
6. **Duplicate Identification Using Supervised Machine Learning**
   ML classifiers distinguish duplicate from non-duplicate pairs based on similarity features:
   - Random Forests: Ensemble trees that handle feature heterogeneity and reduce overfitting.
   - Support Vector Machines (SVM): Effective for high-dimensional similarity vectors.
   - Gradient Boosting Machines (e.g., XGBoost): Powerful for capturing complex feature interactions.
   - Deep Learning (Siamese Networks): Learn similarity functions directly by embedding pairs into a joint feature space to assess closeness.
     These models are trained on labeled datasets of duplicate/non-duplicate record pairs.
7. **Duplicate Elimination via Learning-to-Rank Models**
   Once duplicates are clustered, elimination selects the best representative record using ranking algorithms such as:
   - Gradient Boosted Decision Trees (GBDT): Models ranking criteria (completeness, recency, source trustworthiness) to score candidates.
   - RankNet or LambdaMART: Neural and tree-based ranking models trained on annotated data to optimize record selection.
     These ML-based approaches improve over heuristic rules by learning complex selection patterns.
8. **Storage and Integration with ML Pipelines**
   Deduplicated data and related metadata (confidence scores, provenance) are stored in scalable systems. Feature stores and model registries enable continuous retraining and inference, ensuring up-to-date duplicate detection.
9. **Monitoring and Adaptive Learning in Real-Time Environments**
   Real-time deduplication pipelines leverage:
   - Reinforcement Learning Agents: Adapt thresholds and blocking strategies dynamically based on feedback, optimizing precision and recall.
   - Online Learning Algorithms (e.g., Incremental Clustering, Streaming Random Forests): Continuously update models to handle concept drift and evolving data patterns without retraining from scratch.
     These adaptive ML techniques ensure sustained accuracy in dynamic data streams.

This comprehensive methodology leverages a combination of supervised classifiers, ranking models, clustering algorithms, and adaptive learning approaches to achieve scalable, accurate, and intelligent duplicate detection and elimination in big data environments.

Conclusion

This paper presents a robust and scalable framework for duplicate detection and elimination in large-scale, heterogeneous data environments, with a strong emphasis on the integration of machine learning techniques. By combining advanced tokenization methods with supervised, unsupervised, and reinforcement learning models, the framework effectively identifies both exact and near-duplicate records, adapting dynamically to diverse and evolving data characteristics.

Machine learning enables the system to learn complex similarity patterns, optimize threshold settings, and improve elimination decisions beyond traditional rule-based approaches. The incorporation of online and reinforcement learning supports continuous adaptation in real-time streaming environments, addressing challenges posed by concept drift and data variability.

Scalable blocking and indexing, combined with ML-driven feature extraction and classification, significantly reduce computational overhead while maintaining high accuracy, making the solution suitable for petabyte-scale datasets in big data contexts. This integration enhances data quality, reduces redundancy, and optimizes analytics performance, providing reliable and timely insights for critical sectors such as healthcare, finance, and retail.

In summary, the proposed ML-powered deduplication framework offers a flexible, intelligent, and efficient solution for maintaining data integrity and supporting advanced analytics in dynamic and large-scale data ecosystems.

References

1) Elmagarmid, A. K., Ipeirotis, P. G., & Verykios, V. S. (2007), **Duplicate record detection: A survey,** *IEEE Transactions on Knowledge and Data Engineering*, 19(1), 1–16. https://doi.org/10.1109/TKDE.2007.250581

2) Kejariwal, A., Roy, S., & Srivastava, D. (2012), **An iterative and scalable data cleaning framework using Hadoop,** In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (pp. 1013–1016).

3) Jebamalar, J. T., & Saravanan, V. (2008). A Unified Framework and Sequential Data Cleaning Approach for a Data Warehouse. *International Journal of Computer Science and Network Security*, 8(5), 117–121.

4) Jebamalar, J. T., & Saravanan, V. (2009). Detection and Elimination of Duplicate Data Using Token-Based Method: A Clustering Based Approach. *International Journal of Dynamics of Fluids*, 5(2), 145–164.

5) Jebamalar, J. T., & Saravanan, V. (2010). Token-Based Method of Blocking Records for Large Data Warehouse. *Advances in Information Mining*, 2(2), 5–10.

6) Jebamalar, J. T., & Saravanan, V. (2010). An Evaluation on Current Research Trends in Data Cleaning on Data Warehousing. *The International Journal of Computational Intelligence Research*, 6(3), 405–430.

7) Jebamalar, J. T. (2011). Handling Duplicate Data in Data Warehouse for Data Mining. *International Journal of Computer Applications*, 15(4).

8) Christen, P. (2012). Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. *Springer*.

9) Chaudhuri, S., Ganjam, K., Ganti, V., & Motwani, R. (2003). Robust and Efficient Fuzzy Match for Online Data Cleaning. *Proceedings of SIGMOD*, 313–324.

10) Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). Mining of Massive Datasets. *Cambridge University Press*.

11) Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

12) Zhang, X., et al. (2020). *Reinforcement Learning for Adaptive Entity Resolution in Streaming Data*. IEEE Transactions on Knowledge and Data Engineering, 32(9), 1775–1788.

13) Papadakis, G., Skoutas, D., Thanos, E. & Palpanas, T. Blocking and filtering techniques for entity resolution. ACM Comput. Surv. *53(2), 1–42 (2020).*

14) Wu, X., Li, H., Yoshioka, N., Washizaki, H., & Khomh, F. (2023). Refining GPT-3 embeddings with a Siamese structure for technical post duplicate detection. *arXiv preprint arXiv:2312.15068*.

15) Hosseinzadeh, M. & Azhir, E. *Data Cleansing Mechanisms and Approaches for Big Data Analytics: A Systematic Study* 1–14 (Springer, 2021).

16) 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), 1–12 (2022).

17) Malik, G., Yildirim, S., Cevik, M., Bener, A., & Parikh, D. (2023). Transfer learning for conflict and duplicate detection in software requirement pairs. *arXiv preprint arXiv:2301.03709*.

18) Feng, S., Suo, W., Wu, Y., Zou, D., Liu, Y., & Jin, H. (2023). A parallel deep learning-based code clone detection model. *Journal of Parallel and Distributed Computing*, 181, 104747.

19) Cho, K., Kamath, G., Larochelle, H. & Murray, N. SemDeDup: Data-efficient learning at web-scale through semantic deduplication. Under review as a submission to TMLR, pp. 1–41. (2023)

20) Toma: A Simple Token-Based Approach for Effective Code Clone Detection. (2024). *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*.