IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 31, No. 4, April, 2026, pp. 1087-1096
Journal homepage: http://iieta.org/journals/isi

Integrating Semantic Transformation Within Extract-Transform-Load Pipelines: A Big Data Perspective

K V Phanikanth*, H S Guruprasad

Department of Information Science and Engineering, B.M.S. College of Engineering, Bengaluru 560019, India

Corresponding Author Email: phani.kv83@gmail.com

Copyright: ©2026 The authors. This article is published by IIETA and is licensed under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/).

https://doi.org/10.18280/isi.310406

Received: 1 October 2025
Revised: 9 December 2025
Accepted: 15 December 2025
Available online: 30 April 2026

Keywords:
data warehouse, Extract-Transform-Load, semantic-ETL, big data analytics, data heterogeneity, 4Vs, business intelligence

ABSTRACT

The rapid expansion of software, the internet, mobile, and distributed computing has made big data analytics indispensable across corporate, public, and industrial sectors, enabling interoperable data-driven decision systems. However, integrating and analyzing heterogeneous data sources remains challenging as traditional Extract-Transform-Load (ETL) processes struggle to meet big data's 4V requirements (Volume, Velocity, Variety, Veracity). To address these issues, we propose a novel big data ETL framework called Semantic Transformation-assisted Correlative Multi-Parametric Duplication Removal (STCMDR), which improves data warehousing efficiency and data quality in relation to all four V dimensions. The framework employs logical data identification and rule-based parsing to extract and unify raw data from diverse sources. Next, the framework applies Word2Vec embedding to convert the integrated data into a low-dimensional semantic vector space, followed by a Correlative Multi-Parametric Duplication Removal (CMDR) phase. CMDR uses cross-correlation analysis and the Wilcoxon rank sum test to identify and remove redundant or less significant entries, retaining an optimal subset of semantic features for loading into storage. To ensure veracity, Semantic Transformation-assisted Correlative Multi-Parametric Duplication Removal-Extract-Transform-Load (STCM-ETL) integrates a data-quality module that applies Synthetic Minority Over-sampling Technique and Edited Nearest Neighbor (SMOTE-ENN) resampling, normalization, and machine learning classification, thereby bolstering data reliability for downstream big data analytics. Simulation results demonstrate that STCM-ETL significantly reduces feature dimensionality, processing latency, and memory usage while maintaining high accuracy. These improvements confirm the framework's robustness and suitability for real-time big data applications.

1. INTRODUCTION

In recent years, digital data generation has exploded across industries due to the rapid growth of internet services, social media, mobile devices, and decentralized applications [1, 2]. This deluge of information is commonly characterized by the "4Vs" of big data: Volume, Variety, Velocity, and Veracity [3]. In practical terms, organizations are confronted with massive, fast-growing datasets that are often highly heterogeneous (varying formats and structures) and generated continuously in real time [4-7]. Managing and extracting value from such data is challenging, yet crucial – modern enterprises rely on data-driven decision systems in domains like e-commerce, social media analysis, and healthcare, which demand continuous data processing and analysis [8-10]. The inherent value in these large-scale datasets makes it imperative to store, integrate, and analyze the data despite the difficulties in handling their size and complexity [3-6, 8].

To harness big data for analytics, organizations typically consolidate data from multiple sources into a data warehouse – a centralized repository designed for query processing and decision support [11, 12]. Extract-Transform-Load (ETL) processes are the backbone of data warehousing [8, 12]. In an ETL pipeline, data is extracted from diverse source systems, transformed into a unified schema (e.g., a multi-dimensional data cube format) through cleaning and restructuring, and then loaded into the warehouse for future analysis [8, 11, 12]. Effective ETL is critical: analytical tools and on-line analytical processing (OLAP) systems can only deliver reliable insights if the underlying data is well-structured, clean, and readily accessible [8]. ETL enables moving data from operational databases to analytical repositories, making integrated historical data available for business intelligence and interoperability across cloud or distributed environments [11, 12]. However, conventional ETL pipelines struggle with big data. Despite their importance to big data analytics [13], traditional ETL methods often become time-consuming and cost-prohibitive when dealing with extremely large, heterogeneous inputs [14]. The complexity of extracting and transforming data from numerous sources at scale can overwhelm existing ETL tools, leading to bottlenecks in data warehousing.

A number of research efforts and commercial tools have attempted to improve ETL performance under big data

constraints, but significant limitations remain. Academic approaches: Many studies focus on optimizing the transformation phase (e.g., data cleaning, filtering, merging) to streamline ETL [8, 11, 12]. While these techniques can accelerate certain tasks, generalizing them to arbitrary heterogeneous data sources remains difficult [8]. Some works introduce semantic or ontological methods to enrich and speed up transformations [8-17], yet these often incur high computational overhead or latency [8-13]. To support real-time analytics, researchers have also explored parallel processing frameworks—Hadoop MapReduce [17] and Apache Spark [18]—to expedite data loading. These frameworks improve throughput, but they come with trade-offs: Hadoop's distributed storage and processing can introduce substantial computational overhead [16-18], and Spark (while faster) does not inherently ensure data completeness or accuracy (addressing Volume and Veracity) [16-18]. Industry tools: Modern ETL software (open-source platforms like Pentaho Kettle, Talend [19, 20], and commercial suites like Informatica, SAS Data Management, Oracle Data Integrator, IBM DataStage [21-23]) provide user-friendly GUIs and flexibility in designing workflows. Despite their usability, these tools still struggle with the 4V challenges at extreme scale [8, 24]. They excel at mapping data between sources and targets, but issues like complex data heterogeneity, high-speed streaming input, and data quality assurance are only partially addressed. For example, some ETL configurations include duplicate record removal and data merging to reduce data volume, but this can lead to false positives (erroneous deletions) under typical big data conditions such as imbalanced, high-dimensional data with complex, non-linear patterns [25]. Simply dropping data to save space may improve memory usage, yet it risks compromising analytical accuracy (Veracity) if important information is lost [8]. Balancing efficiency (Volume/Velocity) with information preservation (Veracity) remains an open challenge in current ETL solutions.

Table 1. Limitations of existing Extract-Transform-Load (ETL) approaches

Big Data Constructs	ETL Motive
Volume	Traditional ETL is time- and resource-intensive when handling very large datasets, often failing to scale without significant delays or cost [8, 14].
Variety	Many tools cannot easily accommodate heterogeneous, unstructured, or high-dimensional data; extensive manual mapping or custom code is required to integrate diverse sources [8, 24].
Velocity	Real-time or near-real-time data ingestion is problematic for most ETL systems; they often cannot keep up with rapid data streams or require expensive infrastructure (e.g., large Hadoop/Spark clusters) to do so [16-18].
Veracity	Ensuring data quality and accuracy is difficult; performance optimizations (like aggressive filtering or distributed processing) sometimes sacrifice reliability and consistency in the data, undermining trust in the analytics [8, 25].

Given these limitations as shown in Table 1, there is a clear research gap: current ETL methodologies do not fully meet big data requirements. No existing solution provides a generalized, scalable ETL pipeline that simultaneously handles huge volumes, diverse data types, high ingestion rates, and stringent accuracy needs [8]. Integrating data quality optimization into each stage of ETL (to maintain Veracity while addressing Volume, Variety, and Velocity) is largely unexplored in the state-of-the-art approaches. This gap forms the motivation for our work.

To address these challenges, we propose a novel ETL framework called Semantic Transformation assisted Correlative Multi-Parametric Duplication Removal-Extract-Transform-Load (STCM-ETL). The STCM-ETL approach augments the standard extract–transform–load pipeline with a data quality optimization layer, targeting all 4Vs of big data. In the proposed framework, input from multiple heterogeneous sources (e.g., CSV, JSON, XML files) is first logically parsed and extracted using rule-based mappings to ensure no significant information is lost (addressing Volume and Variety). The data is then subjected to semantic transformation: we employ word embedding techniques (e.g., Word2Vec) to represent textual and unstructured data in a consistent feature space, enabling the integration of diverse data types and reducing dimensionality. Next, a Correlation-based Multi-Parametric Duplication Removal (CMDR) algorithm operates on the transformed data to identify and eliminate redundant records across incremental batches. This step uses statistical measures (such as cross-correlation and significance testing) to drop duplicate or highly correlated entries, minimizing data volume while preserving critical information (thereby improving processing Velocity and maintaining Veracity). The refined, cleaned data is then loaded into the data warehouse for analysis. To further ensure high data quality, the framework incorporates additional measures like resampling (to handle class imbalances), normalization, and even a lightweight machine learning validation step that checks if predictive accuracy remains high after ETL. Through this integrated design, STCM-ETL addresses the full spectrum of big data ETL challenges: it is built to scale with large volumes, accommodate variety via semantic integration, ingest streaming data efficiently, and uphold veracity by filtering out noise and enforcing data quality standards.

The novelty of STCM-ETL lies in its combination of semantic transformation and correlation-driven data reduction within the ETL pipeline to tackle big data's core challenges holistically. This is, to our knowledge, one of the first ETL solutions to cumulatively address Volume, Variety, Velocity, and Veracity requirements in a unified framework. Experimental results demonstrate that the proposed approach can drastically improve performance and reliability: for example, our prototype achieved about 93% reduction in data storage requirements while maintaining ~98% accuracy (95% F-measure) on analytics tasks, confirming that significant efficiency gains are possible without sacrificing data quality. Thus, STCM-ETL fills the identified research gap by providing a robust, scalable ETL strategy for real-time big data analytics, ensuring that enterprises can derive timely and trustworthy insights from their rapidly growing data assets.

2. RELATED WORK

ETL architecture and frameworks: Many efforts have focused on designing robust ETL pipelines for big data environments. Traditional ETL tools and earlier frameworks [13] were effective for structured data but struggled with the scale and heterogeneity of modern datasets. Some works explored user-friendly GUI-based ETL tools [15] to simplify

development, though their effectiveness on real-world big data remained limited. Recent research proposes new ETL architectures tailored for volume and velocity: For example, an Extract–Clean–Load–Transform (ECLT) pipeline optimizes unstructured text handling by cleaning data before loading [2], and a MapReduce-based ETL (ETLMR) leverages Hadoop to speed up loading and transformation for large warehouses [16]. Other frameworks integrate distributed processing and streaming capabilities, such as an on-demand streaming ETL model that partitions data via distributed in-memory caches and serverless pipeline designs that trade off scalability and cost across cloud services [7]. Domain-specific implementations further demonstrate ETL's role in analytics, from human resources data integration [1] to financial data warehouses for trading insights [9]. These architecture-centric approaches improve ETL throughput and scalability, but they often focus on performance aspects; challenges like seamless heterogeneous data integration and minimizing manual effort in schema/metadata management persist.

Semantic ETL approaches: To address data variety and meaning, some researchers incorporate semantic techniques into ETL. One theoretical model extends the pipeline to "E(G)TL" by adding a generative AI-based *Generate* step, enriching and profiling data during transformation [3]. Others employ semantic web standards, for instance storing integrated data in RDF format and using SPARQL for flexible querying [17]. A number of studies advocate ontology-driven or semantic ETL models [22], aiming to better merge heterogeneous sources using knowledge bases. It has been observed that classical ETL struggles with highly heterogeneous big data, motivating semantic methods for more adaptive integration [18]. However, many semantic ETL solutions remain conceptual or semi-manual. For example, an ontology-based integration approach achieved richer data merging [22] but relied heavily on manual mapping of knowledge, limiting its scalability for high-volume, fast-changing data. Overall, while semantics promise more intelligent data unification, existing methods have yet to fully automate the process or demonstrate efficiency at big data scale.

Data quality and cleansing in ETL: Ensuring data quality during ETL is crucial for reliable analytics, and several frameworks embed cleansing operations into the pipeline. One extensive framework integrates anomaly detection into ETL using predictive analytics to catch data quality issues (e.g. outliers, inconsistencies) before loading [5]. Another approach adds a dedicated deduplication stage to identify and remove redundant records across multiple sources, using a six-step record linkage method (implemented as a DataCleaner tool extension) to enforce a single "golden record" [8]. Some research also emphasizes the need for adaptive multi-source integration to maintain quality; for instance, a systematic analysis-based ETL method was proposed to consolidate heterogeneous data streams, but it noted that automating this integration is necessary to handle evolving sources. These quality-focused solutions improve the consistency and reliability of data entering warehouses. Nonetheless, they often come at the cost of additional computation or complexity, and they typically tackle specific issues (like duplicates [8] or anomalies [5]) without addressing broader semantic consistency. Integrating such data quality techniques seamlessly, especially under real-time constraints, remains an open challenge.

ML-based ETL and automation: Researchers have increasingly applied machine learning to automate and optimize parts of the ETL process. Advanced extraction techniques use AI to handle unstructured data—for example, NLP and deep learning models now parse documents such as invoices to extract structured fields in place of manual rules [4]. Machine learning has also been used to enhance performance and decision-making in ETL pipelines. A hybrid meta-heuristic approach combined Grey Wolf optimization and Tabu Search to streamline transformation, reducing redundant processing and improving load balancing in a cloud data warehouse ETL workflow [6]. Surveys of emerging ETL pipelines note a trend toward ML-driven orchestration and heuristics for serverless and big data contexts [7]. In addition, AI has been embedded into transformation stages: the earlier mentioned generative ETL model uses AI to enrich data [3], and predictive models within ETL can flag poor-quality data for removal [5]. These ML-based approaches show potential for reducing manual effort and handling complex data patterns. However, they are often tailored to specific tasks or domains (e.g., document extraction [4] or particular optimization targets [6]). Deploying a general, end-to-end intelligent ETL solution that dynamically handles variety, scale, and data quality issues remains difficult due to the complexity of training and integrating such models into all ETL phases.

In summary, existing work on ETL for big data addresses individual aspects but still exhibits important gaps. Many approaches emphasize performance and scalability (e.g., parallel loading with MapReduce [16]) or ease of use (GUI mappers [15]), yet they often neglect deeper automation for heterogeneous data integration. Techniques that improve data quality or semantic consistency are usually isolated—for instance, anomaly detection [5] or deduplication [8] frameworks handle specific issues but are not combined with semantic transformation. Meanwhile, semantic ETL models [22] and AI-driven proposals [3] have not overcome the need for manual setup or demonstrated generalizable, real-time performance. As a result, no single solution currently ensures high throughput, automatic semantic integration, and rigorous data quality handling altogether. The proposed STCM-ETL approach is designed to fill this gap by integrating semantic transformation with automated multi-source duplicate removal and optimization. By doing so, STCM-ETL seeks to simultaneously tackle the Volume, Variety, Velocity, and Veracity challenges of big data ETL, offering a more comprehensive and novel solution compared to prior work.

3. PROBLEM FORMULATION

Efficient big data analytics depends critically on the quality, relevance, and structure of data produced during the ETL pipeline. However, existing ETL approaches predominantly optimize isolated stages—either extraction (rule-based parsing and cleansing), transformation (semantic or rule-based mapping), or loading (Hadoop/Spark-based distribution). This fragmented optimization leads to suboptimal outcomes because data redundancy, low-significance attributes, and heterogeneous formats continue to propagate through the pipeline, degrading analytic accuracy, increasing storage cost, and inflating computation time. Despite several attempts at feature reduction and duplication elimination, current models do not adequately examine how such reductions affect the veracity and reliability of analytical outputs.

A clear research gap exists: no unified ETL framework simultaneously performs semantic transformation, multi-parameter duplication removal, and data-quality optimization while preserving veracity for incremental, multi-source data streams. Existing methods either reduce data without assessing its analytical significance or load cleaned data without verifying that the remaining subset is both computationally efficient and analytically reliable. This calls for an integrated ETL strategy that treats extraction, transformation, loading, and quality assurance as a cohesive pipeline rather than independent tasks.

3.1 Core limitations in existing Extract-Transform-Load models

- Focus on isolated stages (extraction-only or loading-only optimization) rather than an end-to-end integrated pipeline.
- Redundancy removal methods do not assess the analytical significance or contribution of retained features.
- Semantic transformation techniques reduce dimensionality but do not ensure preservation of data veracity.
- Loading frameworks (e.g., Hadoop/Spark) operate blindly on whatever data arrives, without built-in quality or relevance checks.
- Lack of support for incremental, multi-source, and heterogeneous data ingestion with reliability guarantees.
- Insufficient evaluation of how data reduction impacts downstream analytics accuracy and stability.

3.2 High-level architecture of the proposed solution

To address these gaps, the proposed STCM-ETL architecture integrates:

- Rule-based multi-source extraction: logical parsing, heterogeneity handling, and unified sequence generation.
- Semantic transformation: tokenization followed by Word2Vec-based embedding to obtain low-dimensional semantic vectors.
- CMDR: eliminating duplicates and low-significance elements using correlation and statistical ranking tests.
- Incremental & interval-based loading: controlled warehouse population with reduced data size and latency.
- Data-quality optimization layer: using Synthetic Minority Over-sampling Technique and Edited Nearest Neighbor (SMOTE-ENN), normalization, and ML-based validation to ensure reliability and veracity of stored data.

4. RESEARCH QUESTIONS

This study formulates a set of focused research questions intended to establish the foundation for a unified and reliable ETL framework for big data analytics.

4.1 Research question 1

Can logical and rule-based extraction methods effectively retrieve and structure attributes from heterogeneous data sources?

Justification: Heterogeneous formats (CSV, JSON, XML, text streams) remain a core bottleneck in ETL pipelines. Determining whether rule-based parsing truly improves consistency and interoperability is essential for scalable extraction.

4.2 Research question 2

Can Word2Vec-based semantic embedding enable cost-efficient and delay-resilient transformation for big data ETL?

Justification: Semantic embeddings may reduce dimensionality and transformation complexity. Assessing their impact on execution time and resource utilization is critical for high-volume environments.

4.3 Research question 3

Can CMDR improve ETL reliability by eliminating redundant or low-significance attributes without compromising data veracity?

Justification: Duplication removal affects storage, latency, and analytic accuracy. Understanding whether CMDR preserves meaningful information is central to robust ETL optimization.

4.4 Research question 4

Does integrating extraction, semantic transformation, and duplication removal within a unified pipeline enhance overall ETL performance and downstream analytics reliability?

Justification: Instead of evaluating components in isolation, the combined influence of these techniques must be assessed to confirm whether an integrated architecture yields measurable improvements in accuracy, efficiency, and data quality.

5. SYSTEM MODEL

5.1 Overview of Semantic Transformation assisted Correlative Multi-Parametric Duplication Removal-Extract-Transform-Load framework

We propose a Semantic Transformation assisted STCM-ETL framework tailored for big data. The system performs multi-level optimization across the ETL pipeline (Extraction, Transformation, Loading) to address the 4Vs of big data (Volume, Variety, Velocity, Veracity). Figure 1 illustrates the overall architecture (multi-source input, transformation, and optimized loading). In summary, STCM-ETL first integrates data from heterogeneous sources, then applies cleaning and semantic embedding to reduce data volume, followed by the novel CMDR mechanism to eliminate redundant information while preserving important features. Finally, an incremental loading strategy stores the optimized data, and an optional analytics module ensures the veracity of the outcomes. The key innovative components – Word2Vec-based embedding, CMDR (correlation analysis with a Wilcoxon rank sum significance test), and incremental loading – work in concert to improve ETL efficiency under high Volume (large data), Variety (multiple formats), Velocity (streaming inputs), and Veracity (data quality). Below, we detail each stage of the

proposed method and its contributions. The proposed ETL model was executed, as depicted in Figure 1.

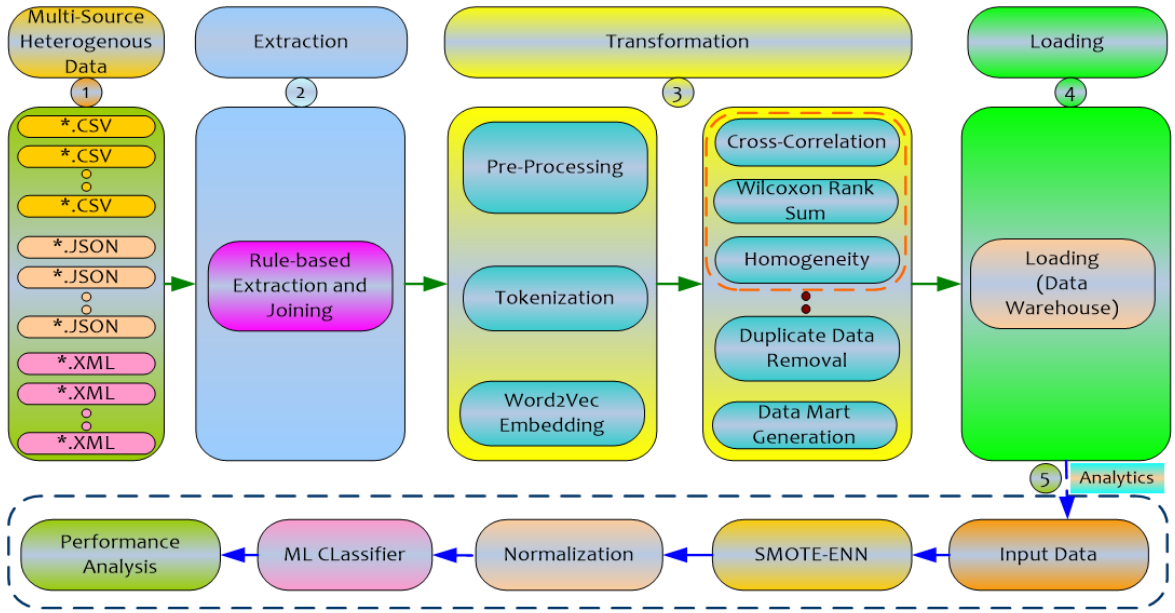

Figure 1. Proposed STCM-ETL framework
Note: STCM-ETL = Semantic Transformation assisted Correlative Multi-Parametric Duplication Removal- Extract-Transform-Load.

5.2 Extraction phase

Multi-source heterogeneous data extraction: To handle variety, STCM-ETL supports multiple file formats (e.g., CSV, JSON, XML). We employ format-specific parsing and logical rules to extract and unify data from these sources. For instance, an XML parser (Python's xml.etree module) is used to convert XML files into structured elements (tags, attributes, text) for further processing. Similarly, JSON and CSV inputs are parsed using appropriate routines. All extracted data fields are then joined/merged into a consolidated dataset. This flexible extraction mechanism allows the framework to ingest 15 sequential data chunks (in our experiments, 5 in each format CSV/JSON/XML) which emulate incremental real-world feeds. By integrating disparate sources in a streaming fashion, the extraction phase addresses Variety and contributes to Velocity (through continuous, batched ingestion).

5.3 Transformation phase

After extraction, the unified data undergoes a series of transformation steps to reduce noise, generate semantic features, and remove duplicates:

5.3.1 Pre-processing and tokenization

We apply standard text preprocessing to normalize the data. This includes removing punctuation and other non-word symbols, eliminating stop words (using NLTK stop-word lists), and converting text to lower-case. These steps strip out noise and ensure consistency without losing informational content. Next, each pre-processed text entry is tokenized into words. By splitting sentences into individual word tokens, we prepare the data for vectorization. (These basic cleaning steps are well-known in ETL; hence we summarize them briefly.)

5.3.2 Semantic embedding with Word2Vec

To tackle data volume and produce a machine-readable representation, STCM-ETL transforms tokens into semantic embeddings using Word2Vec. We leverage Google's pre-trained Word2Vec model (300-dimensional vectors) to encode each word token into a dense numeric vector. This yields a low-dimensional feature matrix for the input corpus, dramatically reducing data size while preserving meaningful relationships between words. In our implementation, each token is mapped to a latent vector, and token context (neighboring words within a window) influences the embedding. The result is an embedding matrix that captures semantic similarities. By compressing text into numeric vectors, this step decreases storage and memory needs (addressing Volume) and can speed up processing (Velocity) due to the smaller feature space. It also provides a unified feature representation across heterogeneous sources, aiding Variety handling.

5.3.3 Correlative multi-parametric duplication removal

This is the core novel contribution of our method, aimed primarily at reducing redundant data volume and improving processing velocity without compromising data veracity. CMDR operates on the Word2Vec embeddings to detect and remove duplicate or highly correlated data points across the incremental input streams. It consists of two complementary techniques: a cross-correlation analysis to find homogeneous (highly related) features, and a Wilcoxon rank sum test (WRST) to evaluate each feature's significance. The motivation is that simply dropping all strongly correlated data might remove information that is still important for predictive accuracy. Thus, CMDR filters out only those features that are both duplicated and insignificant.

5.3.4 Correlation-based redundancy check

We compute pairwise correlations among the semantic feature vectors using Pearson's correlation coefficient (Eq. (1)). If the correlation between two data vectors exceeds a threshold (set to 0.6 in our design), they are considered homogeneous duplicates. High correlation (values near 1) indicates that one feature is largely redundant given the other. Such duplicated elements can increase storage and computation costs without benefiting the analysis [10]. We

mark all features with ≥ 0.6 correlation for potential removal to reduce the data volume. This correlation-driven filtering ensures only one representative from any group of highly similar features is retained, thus cutting down data repetition.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})\sum_{i=1}^{n}(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (1)$$

5.3.5 Significance filtering with Wilcoxon Rank-Sum Test (WRST)

Rather than dropping all highly-correlated items, we introduce a significance test to preserve those that are still informative. We apply the Wilcoxon Rank-Sum Test (WRST)– a non-parametric test on paired samples – to assess each feature's importance (contribution to the target outcome). WRST evaluates whether the distribution of a feature's values significantly differs with respect to the target class labels (in our case, sentiment classes). In practice, we compute a signed rank for each feature and derive a test statistic (T) to measure significance (Eq. (2)). Features yielding high WRST scores are deemed significant predictors for the analytics task (e.g., sentiment classification), whereas low-score features are considered insignificant. By doing this, even if a feature is highly correlated (duplicated) with others, it will not be removed if it carries unique predictive significance. This two-tiered strategy (correlation + WRST) balances Volume/Velocity optimization with Veracity: we eliminate truly redundant data but retain duplicates that are valuable for accuracy.

5.3.6 Correlative Multi-Parametric Duplication Removal algorithm

The pseudocode in Algorithm 1 summarizes how correlation and WRST are combined in CMDR to determine the final set of features to keep. Eq. (2) (referenced in the pseudocode) formalizes the rule for retaining data elements that pass either the homogeneity check or the significance check (or both). Essentially, a data item is retained if it is not highly correlated, or if it is correlated but also statistically significant according to WRST.

Algorithm 1. Pseudo-code for CMDR duplicate data removal

Input: Word2Vec embedding metrics for each data source $E_{i=1,2,3}$
Output: Retailed data for loading to the warehouse
Step 1. Input the semantic embedding metrics
Step 2. Calculate homogeneity values by estimating correlation values C_i (1) (say, H_i)
Save the data elements H_i with $C_i > 0.6$
Step 3. Calculate WRST output ($p - value$) for each data (R_i)
Save R_i with $p_i > 0.5$
Step 4. Measure the CMDR-based retainable data as per (3)

$$D_i = \{H_i \cap R_i\} \quad (2)$$

Step 5. Repeat Step 1 to Step 4 to retain the optimal set of data elements D_i, where $i \in \{*.CSV, *.JSON, and *.XML\}$

By applying CMDR on the embedded data, the transformation phase yields an optimal feature set – a reduced dataset containing only non-duplicate, significant features ready for loading. This greatly shrinks the data size (fewer features and samples) and speeds up downstream analytics, addressing the Volume and Velocity challenges. Notably, because CMDR uses WRST to guard against dropping important data, the Veracity (accuracy potential) of the data is preserved. In effect, STCM-ETL's transformation ensures the data warehouse only stores meaningful and non-redundant information from the heterogeneous inputs.

5.4 Load phase

After duplication removal, the loading phase stores the curated data into a target repository (data warehouse). We implement an incremental loading mechanism (Algorithm 2) to update the warehouse with each incoming batch of processed data. Initially, if the warehouse is empty, it simply loads the first batch. For subsequent batches, the new retained data (output of CMDR) is concatenated with the existing warehouse data. This procedure ensures that as new data streams in, only the new unique significant features are added to the warehouse, preventing any re-loading of duplicates. In our implementation, we used a serialized binary format (Python pickle file) as a stand-in for the warehouse for simplicity. The incremental loading strategy supports Velocity, allowing real-time or periodic updates to the stored data without re-processing the entire dataset each time.

Algorithm 2. Incremental loading

Input: Data Stored in prior Data Warehouse, New Data at Source
Output: Updated data in data warehouse
Step 1. Input the New_{data} from input source
Step 2. Perform overall process (Figure 1) to retrieve updated change data metrics or $New_{Embed\_Metrics}$ and retrieve D_{i_New} as per (3)
Step 3. Fetch D_i from the data warehouse
Step 4. if D_i is empty, then update with D_{i_New}

$$D_i \leftarrow D_{i\_New}$$

else, concatenate existing D_i and D_{i_New}

$$D_{Update} = conc(D_i, D_{i\_New})$$

Step 5. Update data warehouse with D_{Update}

5.4.1 Ensuring veracity with data optimization

While the above steps address Volume, Variety, and Velocity, we include an additional computing optimization module to bolster Veracity (data quality and accuracy for analysis). The concern is that even after CMDR, the data might suffer from issues like class imbalance or scale disparities, which could hurt analytics reliability. To mitigate this, the loaded data is passed through a resampling and normalization process before analysis. We apply an integrated SMOTE-ENN resampling technique to balance the class distribution: SMOTE generates new samples for minority classes, and ENN cleaning removes any ambiguous synthetic points. This yields a more balanced dataset, improving the fairness and representativeness of the data for model training. We then perform Min-Max normalization to scale all features to [0,1] range, reducing the effect of differing value scales (Eq. (3)). Finally, to validate that our ETL output indeed maintains high fidelity for decision-making (veracity), we test it with downstream machine learning classification. In our study, we feed the processed data into various classifiers (Naïve Bayes, SVM, Decision Tree, k-NN, Logistic Regression, etc.) to predict the sentiment labels. The classification accuracy and F-measure serve as indicators of data quality – if our ETL preserves veracity, the models should achieve strong predictive performance. This computing optimization stage is

essentially a plug-in analytic layer on top of STCM-ETL: it does not alter the ETL process itself, but it confirms that the ETL's output supports accurate analytics and thereby meets the Veracity criterion of big data. By incorporating this step, the overall framework not only produces smaller and faster-to-process data, but also verifies that the reduced data is still informative and trustworthy for end-use.

The proposed STCM-ETL framework innovatively combines rule-based multi-source extraction, semantic embedding, and the CMDR (correlation + WRST) mechanism to optimize the ETL pipeline for big data. These contributions directly address the challenges of Variety (handling heterogeneous formats), Volume (dimensionality reduction and duplicate elimination), Velocity (faster processing via data reduction and incremental loading), and Veracity (ensuring the retained data maintains predictive quality). By the end of the process, the data warehouse contains a significantly smaller yet information-rich dataset, which can be analyzed more efficiently without sacrificing accuracy.

5.4.2 Experimental setup: dataset

We evaluated the STCM-ETL framework using a benchmark social media sentiment dataset. This dataset consists of approximately 36,000 text reviews (sentences), each labeled with a sentiment (Positive/Negative). After initial feature extraction, each review is represented by a 300-dimensional feature vector (reflecting the Word2Vec embedding size). The corpus was chosen to mimic real-world big data characteristics: it is large-scale (tens of thousands of instances, addressing Volume), contains unstructured text from social platforms (Variety), and can be fed into the ETL system in successive chunks to simulate streaming (Velocity). The source of the data is a public sentiment analysis benchmark (social media reviews), ensuring that results are relevant to practical big data analytics scenarios. We also constructed 15 sequential input files from this corpus (5 each in CSV, JSON, and XML format) to serve as the multi-source input for our ETL pipeline. Each file contains a subset of the reviews, and these were ingested one after the other by STCM-ETL, emulating an incremental data arrival process. This setup tests the framework's ability to handle heterogeneity and continuous data flow, while ultimately preserving the dataset's ground truth labels for evaluating veracity.

5.4.3 Hardware and software configuration

All experiments were conducted in a Python-based environment on a standard computing platform. Below is a summary of the system configuration:

- Hardware: PC with an octa-core Intel processor (Core i7 or equivalent) and 16 GB RAM. No specialized GPU or cluster computing was required, as the focus is on the efficiency of the ETL process itself.
- Software: Windows 10 OS (64-bit) with Python 3.x. Key libraries used include pandas for data handling, xml.etree and json libraries for parsing XML/JSON, NLTK for text preprocessing (stop-word removal, etc.), and Gensim (with Google's pre-trained Word2Vec model) for semantic embedding. The SciPy/NumPy stack was used for correlation calculations and implementing the WRST. For the veracity evaluation stage, we used the scikit-learn library (version 0.24) for SMOTE-ENN resampling (via the imbalanced-learn extension) and various classifier algorithms (Naïve Bayes, SVM, Decision Tree, k-NN, Logistic Regression, etc.).

This configuration ensures that the implementation is reproducible and that performance metrics (memory usage, execution time) are measured in a realistic computing setup.

5.4.4 Baseline Extract-Transform-Load (ETL) models for comparison

To validate the contributions of each component in STCM-ETL, we compared our framework against two baseline ETL strategies:

Baseline 1 – traditional rule-based ETL: This baseline performs conventional ETL operations without any semantic embedding or advanced duplicate removal. It involves extracting from the multiple sources and applying the same basic pre-processing (punctuation removal, stop-words filtering, case normalization, tokenization), but then directly loading all data to the warehouse as-is. In other words, it relies on simple rule-based parsing and cleansing, without reducing feature dimensionality or eliminating duplicates. This represents a typical "syntactic" ETL pipeline that might be used in practice, and serves as a control to measure improvements in Volume and Velocity made by our semantic and CMDR enhancements.

Baseline 2 – semantic ETL without duplication removal: This variant introduces semantic transformation (Word2Vec embedding) but omits the CMDR step. After tokenization, data is converted into 300-dimensional word embedding vectors (just as in STCM-ETL), and all features are then loaded to the warehouse without performing correlation-based filtering. This approach benefits from data compression via embeddings (addressing Volume to some extent), but it does not remove redundant data. We use this baseline to isolate the impact of the CMDR module – by comparing it with STCM-ETL, we can quantify how much duplicate removal further improves storage efficiency and processing speed, as well as any effect on analytics accuracy.

Each ETL model (baselines and the proposed STCM-ETL) is run on the same sequential input dataset for a fair comparison. We track key performance metrics: memory usage (to store the output data) and number of features/records retained after ETL (indicating Volume reduction), ETL processing time per batch (indicating Velocity improvement), and the accuracy/F-measure of a downstream classifier trained on the ETL output (indicating Veracity of the retained data). By analyzing these metrics, the experimental setup directly evaluates how the novel components of STCM-ETL contribute to performance under each of the 4Vs conditions. For instance, improvements in memory and feature count over Baseline 1 highlight the effect of embedding and duplicate removal on Volume; faster loading times highlight Velocity gains; the ability to handle all three file formats demonstrates Variety handling; and maintaining high accuracy and F-measure in sentiment classification confirms that Veracity is not compromised (and is in fact enhanced by the veracity-optimization step).

Overall, this experimental design validates the proposed model by comparing it with simpler ETL pipelines and showing that each added component (semantic transformation and CMDR) yields measurable benefits. The chosen dataset and performance measures align with the 4Vs objectives, allowing us to rigorously assess the STCM-ETL framework's efficacy in a realistic big data context.

6. RESULTS AND DISCUSSION

The proposed STCM-ETL framework was evaluated using a Kaggle social-media sentiment dataset of 32,000 reviews. The data was prepared in CSV, JSON, and XML formats, each requiring logical rule-based parsing. To simulate incremental real-time input, 15 samples (five per format) were generated, totaling 4,80,000 review entries. After applying the STCM-ETL pipeline, the processed data was updated in the warehouse and stored in a data cube.

The framework was implemented in Python and executed on Google Colab using an Intel i5 CPU with 8 GB RAM. Performance was assessed through computation time, memory use, feature and sample reduction, accuracy, and F-measure.

Table 2. Feature retained in data warehouse

Batch Size	Time (s)
3000	79
6000	96
9000	113
12000	142
15000	165
18000	179
21000	190
24000	197
27000	209
30000	221

By eliminating redundant attributes, the framework retained only 147 of the original 300 features, enhancing computational efficiency. Its scalability was further tested across batch sizes ranging from 3,000 to 30,000 samples, with results reported for execution time (Table 2), memory efficiency (Tables 3 and 4), and accuracy and reliability (Tables 5 and 6).

Table 3. Sample retention in data warehouse

Batch Size	Retained Sample
3000	146
6000	353
9000	564
12000	822
15000	1083
18000	1327
21000	1552
24000	1863
27000	2103
30000	2382

Table 4. Memory efficiency

Batch Size	Memory Saving (%)
3000	95.11
6000	94.12
9000	93.73
12000	93.15
15000	92.78
18000	92.63
21000	92.61
24000	92.24
27000	92.21
30000	92.06

Table 5. Accuracy for the proposed STCM-ETL data-model

Batch Size	Accuracy (%)							
	MNB	GNB	BNB	k-NN	SVM	LOGR	DT	Average
3000	95.61	95.77	94.58	96.01	94.53	98.81	98.96	96.32
6000	95.84	94.62	95.01	96.25	95.00	98.81	99.27	96.40
9000	96.98	96.24	96.01	96.33	95.97	99.03	99.11	97.09
12000	96.74	96.21	95.79	96.68	95.56	98.89	98.38	96.89
15000	96.91	96.58	95.93	96.80	96.01	98.93	98.46	97.08
18000	95.86	96.61	96.04	96.54	96.00	99.13	98.52	96.95
21000	96.70	96.73	96.04	95.97	95.79	99.22	98.59	97.00
24000	97.01	96.66	96.38	96.31	96.32	99.05	98.81	97.22
27000	97.73	97.05	96.41	96.68	96.69	99.10	99.07	97.53
30000	96.99	96.94	96.37	97.09	95.94	98.97	99.38	97.38
Average	96.63	96.34	95.85	96.46	95.78	98.99	98.85	-

Note: STCM-ETL = Semantic Transformation assisted Correlative Multi-Parametric Duplication Removal-Extract-Transform-Load.

Table 6. F-Measure for the proposed STCM-ETL data-model

Batch Size	F-Measure (%)							
	MNB	GNB	BNB	k-NN	SVM	LOGR	DT	Average
3000	95.61	94.97	94.82	95.22	93.27	96.07	96.42	95.19
6000	95.61	94.84	95.10	95.14	93.66	96.15	96.71	95.31
9000	95.38	95.05	95.21	94.98	94.01	96.32	96.99	95.42
12000	95.57	95.21	95.33	94.92	94.07	96.58	97.07	95.53
15000	95.61	95.38	95.47	95.04	93.89	96.72	96.89	95.57
18000	95.64	95.52	95.79	95.26	94.13	96.99	97.03	95.76
21000	95.73	95.84	95.77	95.48	93.99	97.00	97.03	95.83
24000	95.77	95.88	95.94	95.77	94.28	97.00	97.67	96.04
27000	95.86	95.71	95.82	95.60	94.59	96.97	97.21	95.96
30000	95.59	95.73	95.99	95.60	94.46	97.96	97.21	96.07
Average	95.63	95.41	95.52	95.30	94.03	96.77	97.02	-

Note: STCM-ETL = Semantic Transformation assisted Correlative Multi-Parametric Duplication Removal-Extract-Transform-Load.

Table 3 shows that execution time increases with batch size: 79 seconds for 3,000 samples, 96 seconds for 6,000, 179

seconds for 18,000, and about 197 seconds for 24,000 samples. The maximum batch size of 30,000 samples required 221 seconds to complete the ETL and loading process.

The proposed ETL framework achieves substantial sample reduction, retaining only essential data across all batch sizes. For example, 3,000 inputs were reduced to 146 samples, 6,000 to 353, and 30,000 to 2,382—yielding about 92–93% memory savings. Similar results across other sizes confirm its strong efficiency for big data analytics.

Table 4 shows consistently high memory savings, ranging from 95.11% for 3,000 samples to about 92% for 30,000 samples. On average, the STCM-ETL framework saves roughly 93% of memory, confirming its strong efficiency in reducing resource usage and processing overhead.

The STCM-ETL framework efficiently supports big data's 3Vs, saving ~93% memory, retaining 147 of 300 features, and enabling faster processing. To ensure analytics performance, the retained data (D_Update) was classified using seven ML models—MNB, GNB, BNB, k-NN, SVM, LOGR, and DT—after SMOTE-ENN resampling and Min–Max normalization. Confusion matrix metrics (TP, FP, TN, FN) were then used to calculate accuracy and F-measure (Table 4).

Table 5 shows that the STCM-ETL framework preserves high sentiment classification performance. Accuracies ranged from 95.78% to 98.99%, with DT achieving the highest at 98.85%, and average accuracy across batch sizes around 97%. F-measure values were similarly high, averaging 95.66% across models (MNB, GNB, BNB, k-NN, SVM, LOGR, DT). These results confirm that, despite reducing over 95% of samples and nearly half the features, the framework maintains sufficient data quality for robust, accurate big data analytics.

The results confirm that the STCM-ETL framework—through sequential parsing, pre-processing, transformation, and optimization—retains an optimal dataset for big data analytics, supporting the 4Vs. Word2Vec embeddings enhance efficiency (RQ2), CMDR reduces data and features for faster, accurate analytics (RQ3), and overall, the framework robustly addresses all research questions (RQ1–RQ4), providing a reliable ETL solution (Figure 1).

7. CONCLUSIONS

This work proposes a robust STCM-ETL for big data applications. It first identifies data source types and formats (CSV, JSON, XML) using logical rule-based methods, extracts and joins the data, and applies pre-processing (stopword removal, lower-casing, tokenization). Word2Vec embeddings convert the joined data into low-dimensional semantic vectors, which are processed using CMDR. WRST and correlation tests select significant items, while duplicates and less relevant data are discarded, reducing memory, computation, and latency, and addressing velocity, volume, and veracity challenges. The retained data is stored in a warehouse or data cube for analytics.

To validate efficiency, the framework incorporates data quality assessment, SMOTE resampling, and machine learning classifiers. Simulations show STCM-ETL achieves ~93% memory savings, ~98% accuracy, and ~95% F-measure, with computation times confirming its suitability for real-time big data applications.

REFERENCES

[1] Tanasescu, L.G., Vines, A., Bologa, A.R., Vaida, C.A. (2022). Big data ETL process and its impact on text mining analysis for employees' reviews. Applied Sciences, 12(15): 7509. https://doi.org/10.3390/app12157509

[2] Farhan, M.S., Youssef, A., Abdelhamid, L. (2024). A model for enhancing unstructured big data warehouse execution time. Big Data and Cognitive Computing, 8(2): 17. https://doi.org/10.3390/bdcc8020017

[3] Vesjolijs, A. (2024). The E (G) TL model: A novel approach for efficient data handling and extraction in multivariate systems. Applied System Innovation, 7(5): 92. https://doi.org/10.3390/asi7050092

[4] Dapkute, A., Siozinys, V., Jonaitis, M., Kaminickas, M., Siozinys, M. (2024). Digital twin data management: Framework and Performance metrics of cloud-based ETL system. Machines, 12(2): 130. https://doi.org/10.3390/machines12020130

[5] Saout, T., Lardeux, F., Saubion, F. (2024). An overview of data extraction from invoices. IEEE Access, 12: 19872-19886. https://doi.org/10.1109/ACCESS.2024.3360528

[6] Widad, E., Saida, E., Gahi, Y. (2023). Quality anomaly detection using predictive techniques: An extensive big data quality framework for reliable data analysis. IEEE Access, 11: 103306-103318. https://doi.org/10.1109/ACCESS.2023.3317354

[7] Dinesh, L., Devi, K. G. (2024). An efficient hybrid optimization of ETL process in data warehouse of cloud architecture. Journal of Cloud Computing, 13(1): 12. https://doi.org/10.1186/s13677-023-00571-y

[8] Phanikanth, K.V., Sudarsan, S.D. (2016). A big data perspective of current ETL techniques. In 2016 International Conference on Advances in Computing and Communication Engineering (ICACCE), Durban, South Africa, pp. 330-334. https://doi.org/10.1109/ICACCE.2016.8073770

[9] Shojaee Rad, Z., Ghobaei-Arani, M. (2024). Data pipeline approaches in serverless computing: A taxonomy, review, and research trends. Journal of Big Data, 11(1): 82. https://doi.org/10.1186/s40537-024-00939-0

[10] Manickam, V., Rajasekaran Indra, M. (2023). Dynamic multi-variant relational scheme-based intelligent ETL framework for healthcare management. Soft Computing, 27(1): 605-614. https://doi.org/10.1007/s00500-022-06938-8

[11] Azeroual, O., Jha, M., Nikiforova, A., Sha, K., Alsmirat, M., Jha, S. (2022). A record linkage-based data deduplication framework with datacleaner extension. Multimodal Technologies and Interaction, 6(4): 27. https://doi.org/10.3390/mti6040027

[12] Biswas, N., Sarkar, A., Mondal, K.C. (2020). Efficient incremental loading in ETL processing for real-time data integration. Innovations in Systems and Software Engineering, 16(1): 53-61. https://doi.org/10.1007/s11334-019-00344-4

[13] Vassiliadis, P. (2009). A survey of extract–transform–load technology. International Journal of Data Warehousing and Mining, 5(3): 1-27. https://doi.org/10.4018/jdwm.2009070101

[14] Seo, Y.S., Huh, J.H. (2024). Artificial intelligence in

finance: Coffee commodity trading big data for informed decision making. IEEE Access, 12: 91780-91792. https://doi.org/10.1109/ACCESS.2024.3409762
[15] Zheng, T., Chen, G., Wang, X., Chen, C., Wang, X., Luo, S. (2019). Real-time intelligent big data processing: Technology, platform, and applications. Science China Information Sciences, 62(8): 82101. https://doi.org/10.1007/s11432-018-9834-8
[16] Liu, X., Thomsen, C., Pedersen, T.B. (2012). Mapreduce-based dimensional ETL made easy. Proceedings of the VLDB Endowment, 5(12): 1882-1885. https://doi.org/10.14778/2367502.2367528
[17] Bansal, S.K. (2014). Towards a semantic extract-transform-load (ETL) framework for big data integration. In 2014 IEEE International Congress on Big Data, Anchorage, AK, USA, pp. 522-529. https://doi.org/10.1109/BigData.Congress.2014.82
[18] Vijayalakshmi, M., Minu, R.I. (2022). Incremental load processing on ETL system through cloud. In 2022 International Conference for Advancement in Technology (ICONAT), Goa, India, pp. 1-4. https://doi.org/10.1109/ICONAT53423.2022.9726039
[19] Gultom, P., Nababan, E.S.M., Mardiningsih, Marpaung, J.L., Agung, V.R. (2024). Balancing sustainability and decision maker preferences in regional development location selection: A multi-criteria approach using AHP and Fuzzy Goal Programming. Mathematical Modelling of Engineering Problems, 11(7): 1802-1812. https://doi.org/10.18280/mmep.110710
[20] Al-Mafrachi, A., Naimi, S. (2024). Losses resulting from exceeding the specified time for completion and its impact on the status of the project and delay in utilizing services. Mathematical Modelling of Engineering Problems, 11(2): 404-412. https://doi.org/10.18280/mmep.110212
[21] Mondal, K.C., Biswas, N., Saha, S. (2020). Role of machine learning in ETL automation. In Proceedings of the 21st International Conference on Distributed Computing and Networking, pp. 1-6. https://doi.org/10.1145/3369740.3372778
[22] Mehmood, E., Anees, T. (2020). Challenges and solutions for processing real-time big data stream: A systematic literature review. IEEE Access, 8: 119123-119143. https://doi.org/10.1109/ACCESS.2020.3005268
[23] Adnan, K., Akbar, R. (2019). An analytical study of information extraction from unstructured and multidimensional big data. Journal of Big Data, 6(1): 1-38. https://doi.org/10.1186/s40537-019-0254-8
[24] Vassiliadis, P., Simitsis, A., Baikousi, E. (2009). A taxonomy of ETL activities. In Proceedings of the ACM twelfth International Workshop on Data Warehousing and OLAP, New York, pp. 25-32. https://doi.org/10.1145/1651291.1651297
[25] Majchrzak, T.A., Jansen, T., Kuchen, H. (2011). Efficiency evaluation of open source ETL tools. In Proceedings of the 2011 ACM Symposium on Applied Computing, New York, pp. 287-294. https://doi.org/10.1145/1982185.1982251