Digital Business 6 (2026) 100167

Contents lists available at ScienceDirect

Digital Business

journal homepage: www.elsevier.com/locate/digbus

LLM-based JSON Mapping and Blockchain Integration for Digital Product Passports☆,☆☆

David Rohrschneider *, Marcel Pehlke, Uwe Handmann, Marc Jansen

Institute of Computer Science, Ruhr West University of Applied Sciences, Lützowstraße 5, Bottrop, 46236, Germany

ARTICLE INFO

Keywords:
Digital Product Passports
Large language models
Blockchain
Data standardization
Industry 4.0

ABSTRACT

The European Commission envisages Digital Product Passports (DPPs) as a mechanism to enable traceable, transparent, and standardized product data across supply chains. This work presents a modular pipeline that transforms raw sensor data into verifiable DPP records using large language models (LLMs) for data standardization and blockchain technology for tamper-proof storage. The system maps unstructured machine-level data to a standardized JSON format and stores it immutably on the Waves blockchain via smart contracts, thereby enabling auditable, machine-readable records suitable for regulatory use. A novel evaluation dataset is introduced to simulate daily production scenarios with varying mapping complexity. The performance of the system is assessed using both proprietary and open-weight LLMs. Results show that the proprietary model achieves the highest accuracy and lowest latency, while open-weight models perform worse as input complexity increases. Multiple prompting strategies were compared, revealing that direct mapping, via few-shot or zero-shot prompts, consistently delivered higher accuracy than approaches based on generating transformation functions. Structured output formatting was also assessed: While it ensured schema validity, it often compromised mapping reliability by introducing incorrect values, likely due to disruptions in model reasoning from output constraints. The proposed architecture demonstrates reliable end-to-end operation with low latency and is suitable for batch-level deployment in real-world production environments. From a practical perspective, the results clarify trade-offs between model choice, prompting strategy, and operational reliability in automated DPP generation. For policymakers, the findings highlight how choices around schema clarity and data granularity shape system design and operational effort in future DPP implementations.

1. Introduction

The increasing adoption of cyber–physical production systems and Internet of Things (IoT)-enabled manufacturing has led to a rapid growth of heterogeneous, machine-generated data in industrial environments (Hashem et al., 2015; Khan et al., 2017). While Industry 4.0 concepts enable fine-grained monitoring through sensor-based data acquisition, they also introduce substantial challenges related to data integration, as machines rely on diverse data models, communication protocols, and vendor-specific semantics (Gleim et al., 2020; Noaman et al., 2022). The lack of scalable mechanisms for semantic harmonization and trustworthy aggregation, therefore, limits the reuse of operational data across production stages and organizational boundaries (Zeid et al., 2019).

Beyond operational optimization, sustainability- and circular-economy-oriented use cases introduce additional data requirements, as ecological attributes must be derived from and linked to machine-level operational data (Adisorn et al., 2021; Sayogo et al., 2015). In this context, the European Commission is pioneering Digital Product Passports (DPPs) as a concrete application requiring structured, verifiable, and lifecycle-oriented product data at an industrial scale. Under regulations such as the Ecodesign for Sustainable Products Regulation (ESPR), selected product groups will be required to carry such passports between 2027 and 2030 (European Commission, 2024a).

DPPs are increasingly discussed as coordination mechanisms for circular supply chains, as collaboration for circular objectives strongly relies on structured data sharing across organizational boundaries (Jensen et al., 2023; Sudusinghe & Seuring, 2022). Recent literature conceptualizes DPPs as product-linked data repositories that allow distributed actors to both contribute and retrieve lifecycle-relevant information

☆ This article is part of a Special issue entitled: 'Industry 4.0 and Supply Chain Management' published in Digital Business.

☆☆ **Acknowledgment and funding sources:** This work has been co-funded by the European Commission and the Ministry of Economy, Innovation, Digitization, and Energy of the State of North Rhine–Westphalia within the projects Circular Performer Emscher-Lippe and easy.dpp.nrw.

* Correspondence to: Lützowstraße 5, Bottrop, 46236, Germany.
E-mail address: david.rohrschneider@hs-ruhrwest.de (D. Rohrschneider).

https://doi.org/10.1016/j.digbus.2026.100167
Received 25 June 2025; Received in revised form 19 February 2026; Accepted 20 February 2026
Available online 24 February 2026
2666-9544/© 2026 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

in support of circular value creation and decision making (Adisorn et al., 2021; Jensen et al., 2023). Such cross-tier collaboration depends on reliable product-level information that accompanies goods through forward and reverse logistics flows, reducing information asymmetries between manufacturers, suppliers, service providers, and recycling actors (Götz et al., 2022; Leising et al., 2018). By linking standardized lifecycle data to individual products, DPPs are positioned as key enablers of traceability, compliance verification, and sustainability reporting across distributed value networks (Langley et al., 2023; Zhang & Seuring, 2024). However, the practical realization of this vision presupposes interoperable and structured data generation at the production level, which remains an operational challenge (Jensen et al., 2023; Porshnev et al., 2019).

Meanwhile, rapid progress in the fields of Large Language Models (LLMs) and blockchain technologies enables new technical possibilities for creating, distributing, and verifying DPPs while ensuring data integrity at scale. Abreu et al. (2025), Dasaklis et al. (2022), Greiner et al. (2024). As new regulations approach and technologies evolve quickly, researching scalable data pipelines becomes increasingly important. Despite growing momentum behind DPPs, data availability and upstream data preparation are repeatedly highlighted as practical hurdles for their implementation at scale (Jensen et al., 2023; Zhang & Seuring, 2024).

In response, this work presents a modular data-mapping pipeline that leverages LLMs to convert raw machine outputs into a predefined JSON Schema consistent with provisional DPP specifications, acknowledging that formal requirements remain under development. The architecture also supports anchoring of results to a blockchain, allowing tamper-proof storage and access control. While the system design incorporates blockchain integration, the primary focus of this study is to evaluate the LLM-based mapping process, specifically its semantic precision, robustness across varying task complexities, and the impact of different prompting strategies.

To evaluate the prototype under conditions that resemble an actual production line, a practical scenario was designed, accompanied by a novel synthetic dataset that encompasses time-series operational data from ten machines on a manufacturing floor. Each sample pairs an *unstructured* JSON source with a corresponding *schema-compliant* target. Three difficulty tiers gradually increase task complexity, including unit conversions (e.g., grams → kilograms), semantic field name matching, and deeply nested structures. Together with the implementation codebase,[1] the dataset was made publicly available[2] to facilitate reproducibility and serve as a reference standard for future research on such mapping systems.

The evaluation focuses on how well state-of-the-art models perform in mapping raw data into the required schema under different prompting conditions (zero-shot, few-shot, and code-generating "mapping functions") and further configuration parameters.

The research questions are defined as follows:

(RQ1) **System architecture:** How can LLM-based mapping and blockchain anchoring be combined to ensure data verifiability, modularity, and secure access control?

(RQ2) **Operational performance:** What latency does the end-to-end pipeline incur under realistic throughput?
Since the proposed pipeline relies on sequential, user-facing blockchain writes, latency directly determines whether interactive data mapping and aggregation remain practically usable.

(RQ3) **Mapping quality and prompting strategy:** How accurately do various LLMs convert heterogeneous machine data into DPP-aligned data structures across complexity levels, given the absence of formally defined DPP requirements, and which prompting strategies perform best?

[1] The full implementation is available at https://github.com/hrw-neurolab/dpp-ai-blockchain

[2] The dataset is available at https://www.kaggle.com/dsv/12220483

Addressing these questions requires a research approach that connects technical feasibility with managerial relevance. This study adopts an artifact-based and experimentally evaluated methodology, in which a functional end-to-end pipeline is implemented and assessed under controlled conditions that simulate an industrial production setting. The evaluation analyzes mapping accuracy and latency across configurations to determine statistically optimal setups for operational deployment. By translating technical performance into decision-relevant insights, the study provides an evidence base for organizations considering LLM-assisted data harmonization and blockchain-based verification in regulated production environments. The resulting contributions are summarized as follows:

- **Integrated end-to-end architecture:** A modular pipeline that combines LLM-based semantic mapping and blockchain anchoring to generate verifiable DPPs from heterogeneous machine data.
- **Schema-driven LLM mapping framework:** A method for converting unstructured machine-generated JSON into a predefined DPP-aligned schema, while acknowledging that regulatory specifications are still under development.
- **Blockchain-based aggregation mechanism:** A smart-contract-based design that assigns machine identities and accumulates production data into tamper-evident DPP records.
- **Systematic empirical evaluation:** A comprehensive analysis of mapping accuracy, structural validity, and prompting strategies across increasing task complexity levels.
- **Public benchmark dataset and implementation:** Release of a synthetic industrial dataset with controlled difficulty tiers alongside the full pipeline implementation to ensure reproducibility.
- **Evidence-based implications:** Derivation of practical recommendations for industrial stakeholders and policymakers.

The remainder of this work is structured as follows: Section 2 provides a structured literature review by outlining the current regulatory landscape for DPPs in the European Union and synthesizing prior research on LLM-based data harmonization and blockchain-enabled traceability, culminating in the identification of key research gaps. Section 3 introduces the overall system architecture, highlighting the integration of LLM-based data mapping and blockchain components. Section 4 details the implementation pipeline, including data flow, prompt engineering, and Smart Contract (SC) logic. To evaluate the proposed solution, Section 5 presents the experimental setup, the dataset creation process, discusses the results, and compares various pipeline configurations through ablation studies. The study concludes in Section 6 by summarizing the main findings, outlining key benefits for businesses, discussing limitations, and providing directions for future research, as well as recommendations for practitioners and policymakers.

2. Literature review and research gaps

This section provides an overview of the current state of the art relevant to the development and implementation of DPPs. It covers three key areas: the evolving regulatory framework that mandates DPPs across sectors, the application of LLMs for semantic data harmonization, and the use of blockchain technology to ensure traceability and data integrity. These perspectives jointly frame the legislative and technological context in which DPP systems are emerging.

2.1. Regulatory landscape for DPPs

The ESPR entered into force in June 2024 and is a key driver of eco-conscious product design. It focuses on a wide range of sectors, including textiles, furniture, tires, mattresses, and telecommunications devices, which are slated for gradual DPP implementation through delegated acts between 2027 and 2030 (European Commission, 2024a). While the regulation sets the tone for implementing a cross-sectoral

DPP, it is only part of a broader legislative landscape, where sector-specific legislation is already translating the high-level concept into concrete compliance rules.

One of the most advanced examples is the new EU Battery Regulation (EU 2023/1542), which introduces a dedicated passport for electric vehicle and industrial batteries, mandatory by February 2027 (European Commission, 2023).

Similarly, building materials and products will also be required to carry a DPP according to the new Construction Regulation (EU 2024/3110) (European Commission, 2024b). The regulation explicitly references the ESPR as the core legislation setting out the required infrastructure and harmonized standards. Although the relevant paragraphs entered into force in January 2025, a schedule for implementing DPPs in the building sector has yet to be established.

While the DPP is part of multiple regulations, the European Commission highlights the goal of harmonized standards across all industries. Furthermore, key parameters mentioned throughout the currently available legal documents include detailed information about the carbon footprint, substances of concern, operational performance, and recycling, with multiple access levels for different stakeholders (European Commission, 2023, 2024a, 2024b). While some data points will be publicly accessible (e.g., for consumers), sensitive data, such as detailed material composition, will be restricted to authorized parties (e.g., business partners or government agencies). This role-based access control approach facilitates the accurate and up-to-date collection of DPP data while protecting confidential business information (Kühn et al., 2025).

The European Commission has not mandated a specific technology platform for storing or delivering DPP data, but emphasizes the importance of a decentralized system in which DPPs are created and distributed by the manufacturing party. The ESPR also suggests that DPP implementations should, where feasible, link to or integrate with existing EU databases such as the European Product Registry for Energy Labeling and the SCIP database for substances of concern (European Commission, 2024a). Likewise, each DPP will be associated with a data carrier (e.g., a QR code or RFID tag) that links the physical and digital counterparts. While legislative measures primarily outline the required data, access rules, and compliance criteria for DPPs, the industry is afforded some flexibility in implementation methods and technologies, as long as they meet the transparency and interoperability goals of the legislation. Adisorn et al. (2021), Kühn et al. (2025).

The concept of DPPs builds on established Industry 4.0 principles, particularly the use of digital twins to represent products and assets throughout their lifecycle. In this context, the Asset Administration Shell (AAS) has been widely referenced in prior research as a suitable technical foundation for DPP implementations, as it provides a standardized, machine-readable digital twin model that supports interoperability, modularity, and lifecycle-oriented data exchange (Abdel-Aty et al., 2025; Jung et al., 2024; Kühn et al., 2025; Pourjafarian et al., 2023).

2.2. LLM-based semantic mapping in industrial data pipelines

LLMs have been shown to support optimization and automation in various supply chain management and business tasks, particularly in information-intensive processes (Simchi-Levi et al., 2026; Srivastava et al., 2024). Building on these capabilities, they are increasingly used to harmonize and standardize heterogeneous data formats. Unlike traditional rule-based data treatment, LLMs can infer semantic equivalences between differently named or structured data fields by leveraging their comprehensive language understanding (Kayali et al., 2024; Santos et al., 2025). For instance, an LLM can recognize that `temperature`, `temp`, and `therm_reading` likely refer to the same concept, and map them to a unified attribute. Such semantic normalization capabilities underpin recent work that applies LLMs to the structuring of heterogeneous sensor-generated data in practice.

Berenguer et al. (2024b) propose an LLM-based approach for improving the reusability of sensor-generated data shared in non-interoperable formats. Using meteorological data as a representative domain, the study converts sensor data extracted from HTML into structured representations such as JSON or XML. The approach is evaluated quantitatively using precision and recall, with GPT-4 achieving a precision of 93.51% and a recall of 85.33%.

The authors extend their work on LLM-based sensor data structuring by addressing data retrieval and semantic comparison (Berenguer et al., 2024a). Heterogeneous sensor data is converted into FAIR-compliant structured formats, and word embedding representations of tabular data are created to enable semantic similarity search across datasets. The system is evaluated by comparing different LLMs on a sensor data retrieval task, achieving a precision of 0.90 and a mean reciprocal rank of 0.94 for the best-performing model.

Hojda (2025) studies the use of LLMs for extracting sensor data from unstructured and semi-structured text sources. Using weather data from forecasting portals, including free-text and tabular representations, the author evaluates a workflow that enforces a strict JSON output structure through prompt design. The approach is assessed using precision- and recall-like matching metrics that quantify the correspondence between source and destination data instances. Results show that GPT-4 achieves the highest extraction accuracy, whereas open-source models such as Llama 3 and Mistral achieve comparable performance when provided with appropriate structural guidance.

Lee et al. (2024) investigate the feasibility of using LLMs for real-time sensor data standardization in seamless positioning environments. The study integrates data from smartphones, IoT devices, and Ultra-Wideband systems and evaluates a fine-tuned LLM-based standardization module. Performance is assessed by the difference between the predicted and actual coordinates, measured in meters. It is quantified via Root Mean Square Error and Mean Absolute Error, with reported values of 0.35 m and 0.25 m, respectively.

Xia et al. (2024) propose an LLM-based system for generating AAS instance models from unstructured textual data, such as technical datasheets, to improve semantic interoperability in digital twins within Industry 4.0. Their approach introduces a semantic node representation to capture the meaning of textual inputs and leverages LLM agents, including Retrieval-Augmented Generation mechanisms, to translate this information into standardized AAS models. The system is evaluated by measuring the effective generation rate of valid AAS elements, yielding values between 62% and 79% depending on the model and configuration.

Shi et al. (2025) address the challenge of mapping proprietary manufacturing data to standardized AAS models by proposing a dual data mapping system based on fine-tuned open-source LLMs. Their approach performs entity matching between existing software data and AAS information models, as well as between AAS models and standardized vocabularies, thereby improving the semantic interoperability of digital twins in Industry 4.0. The system is evaluated through a case study in the injection molding domain and extensive experiments on AAS-specific and general entity-matching datasets, where the fine-tuned LLM achieves state-of-the-art matching accuracy and outperforms GPT-4 with few-shot prompting.

2.3. Blockchain for traceability and verification in DPPs

The growing importance of transparency and accountability in digital business and supply chain operations has led to increased consideration of blockchain technologies, as they provide tamper-resistant storage, decentralized trust, and verifiable data provenance (Esmaeilian et al., 2020; Queiroz et al., 2019; Saberi et al., 2019). In the context of sustainability reporting and product traceability, these properties are particularly relevant because supply chains often exhibit information silos and asymmetries that hinder the verification of product-related claims (Heeß et al., 2024). By recording product information on an

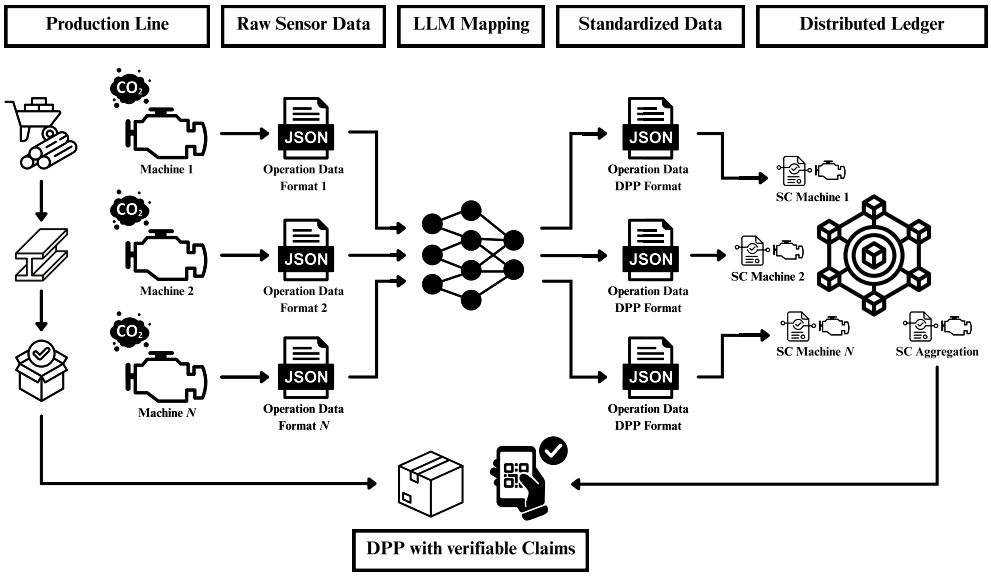

Fig. 1. High-level Architecture: Overview of the modular data pipeline combining sensor-based data collection, LLM-driven data transformation, and blockchain-based storage to generate verifiable DPPs. The architecture reads from left to right, illustrating the data flow from heterogeneous machine outputs to standardized, immutable DPP records.

immutable ledger, blockchain-based systems support traceability and ensure that published data cannot be retroactively altered.

Beyond immutable storage, blockchain systems also enable cryptographic verification of specific product attributes, which is particularly relevant in the context of DPPs. For example, certificates confirming compliance with environmental standards or battery regulations can be referenced in a way that allows third parties to verify their authenticity without accessing the underlying confidential documents. Digital signatures, certificates, and related cryptographic primitives can be anchored on-chain to attest authenticity, certification status, or compliance information, while supporting controlled disclosure of sensitive DPP attributes to authorized parties, such as regulators or supply chain partners, without exposing proprietary production details (Carls et al., 2023; Götz et al., 2022).

Building on these properties, an expanding body of research has investigated blockchain-based architectures for DPPs and related circular economy applications. Prior work demonstrates how blockchain and SCs can support secure data sharing and traceability across supply chains (Christidis & Devetsikiotis, 2016; Voulgaridis et al., 2024), for instance by enforcing predefined rules on how production or recycling data is recorded and shared between independent actors and how blockchain-backed DPPs can improve transparency between manufacturers, regulators, and downstream actors (Kim et al., 2025). These studies establish blockchain as a viable backbone for storing and verifying product-related information in distributed environments.

To support industrial adoption, recent work has also explored permissioned blockchain solutions that integrate with existing enterprise systems. For example, Hulea et al. (2024) implement a DPP architecture based on Hyperledger Fabric, illustrating how permissioned ledgers can provide access control, auditability, and performance guarantees required in industrial contexts.

2.4. Synthesis and research gaps

The reviewed literature shows that LLMs are increasingly applied to semantic mapping of heterogeneous industrial data. Prior work demonstrates that LLMs can extract structured representations, perform semantic retrieval or alignment, and map data into predefined target models such as JSON schemas or AAS representations. In parallel, blockchain-based research on DPPs and related product information systems establishes mechanisms for secure storage, traceability, and verification of structured data. However, this line of work generally assumes that input data is already available in a standardized form and does not address semantic mapping of heterogeneous machine-level data.

Based on this synthesis, the following research gaps can be identified:

1. **Lack of end-to-end pipeline integration:** Existing LLM-based mapping approaches focus on isolated tasks such as extraction, retrieval, or schema alignment, while blockchain-based DPP architectures focus on secure data handling. An integrated, modular pipeline that combines LLM-based semantic mapping with secure aggregation is not investigated.
2. **Indirect alignment instead of direct data transformation:** Most LLM-based studies formulate interoperability as schema matching or entity retrieval against a target model. Direct, end-to-end transformation of heterogeneous machine-operation data into standardized, machine-readable representations has not been explicitly addressed.
3. **Missing support for secure incremental accumulation:** Prior work does not study how semantically mapped data from multiple machines or process stages can be incrementally accumulated with integrity guarantees, for example, via append-only logic enforced through SCs, nor the operational implications (e.g., latency) of such accumulation mechanisms.
4. **Limited evaluation of prompting strategies:** Existing studies typically assess a single interaction paradigm and do not provide a systematic comparison of zero-shot, few-shot, and function- or schema-constrained prompting strategies for reliable structured data generation in industrial mapping tasks.

These gaps highlight the absence of integrated approaches that combine direct LLM-based semantic mapping with secure, incremental data aggregation in industrial settings. Such capabilities are particularly relevant in application contexts such as DPPs, where heterogeneous machine-generated data must be consolidated and verified across product lifecycles.

3. Architecture

The proposed system architecture implements a modular and scalable data pipeline designed to generate DPPs by combining sensor data from a production line, LLM-based data transformation, and a distributed ledger for storage and verification. Fig. 1 presents a high-level overview of the components and data flow, which is read from left to right.

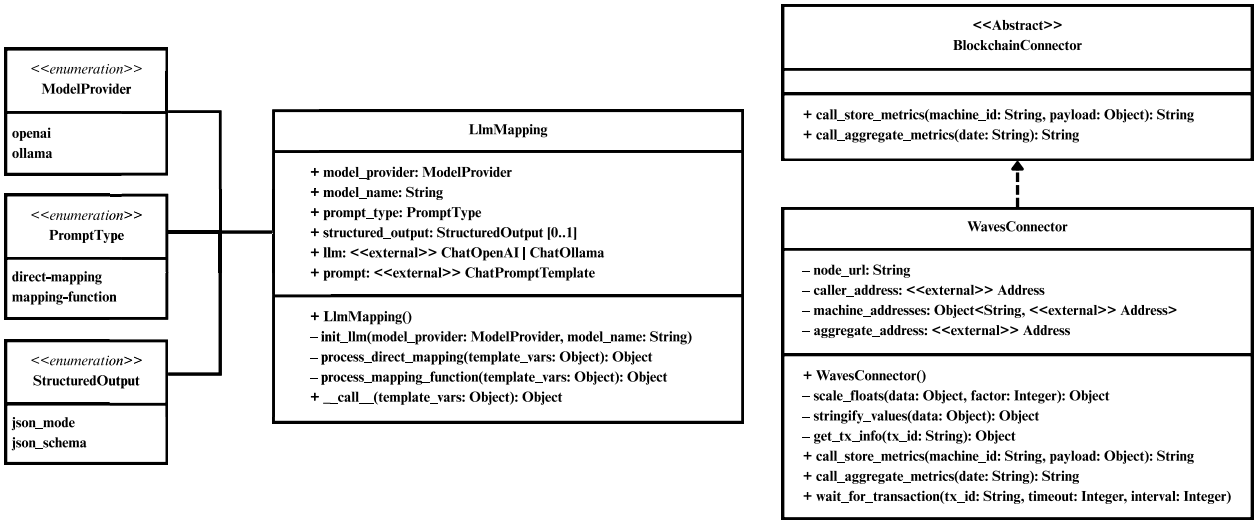

Fig. 2. UML Diagram: Structured visualization of the LLM Mapping and Waves Connector classes with respective attributes and methods.

The following sections will detail each of the components while referring to Fig. 1.

3.1. Production line and raw sensor data

The production line consists of a sequence of machines, $M_1, M_2, \ldots, M_N$, which operate sequentially on the product or its parts, subject to the new regulatory requirements. Each machine embodies a granular production process that emits measurable operational or environmental data. They may vary in type, location, or ownership and are not confined to a single facility. Any process along the production line emitting DPP-relevant data can be integrated into the pipeline. Each machine is equipped with sensors that generate raw data in various machine-specific formats. This data serves as the starting point for building the DPP, as it contains information about environmental impacts that contribute to the overall product carbon footprint. Due to the variety of formats, there is a need for flexible mapping mechanisms that can handle unstructured or semi-structured input. Notably, while this study focuses on JSON-to-JSON mapping, the architecture can be easily adapted to further data types.

3.2. LLM-based data mapping and standardized data

To standardize machine-specific data, we introduce an LLM Mapping mechanism. This component utilizes an LLM to interpret and convert the incoming unstructured or semi-structured sensor data into a predefined DPP-oriented JSON format. The LLM was selected for its flexibility in handling diverse input formats without requiring rigid, predefined parsing rules or domain-specific functions. This adaptability is critical in heterogeneous production environments, where data discrepancies are common, such as inconsistent field naming, differing units, and irregular nesting or encoding practices. By leveraging the LLM's contextual understanding, the system can semantically align diverse input data with a predefined DPP schema.

3.3. Blockchain and smart contracts

Once the data is in a standardized format, it is forwarded to the blockchain layer, where it is recorded in a tamper-evident shared ledger. This ensures that multiple stakeholders can independently verify production-related claims and eliminates reliance on a single central authority to maintain data integrity.

As illustrated in Fig. 1, each machine is associated with a unique address and corresponding SC. These SCs ($SC\ Machine\ 1$ through $SC\ Machine\ N$) receive and store the machine-level DPP data. The choice to assign one contract per machine follows a modular design principle. If new machines are added to the production line or existing ones are removed, the corresponding SC can be deployed or deprecated without requiring changes to other contracts; it only needs to be registered as an entry in the aggregation contract. A single contract would require frequent updates every time the production network evolves. The modular structure ensures that the architecture remains maintainable and flexible over time. An additional SC, $SC\ Aggregation$, collects references to the per-machine data and composes them into aggregated records at the product or batch level. This aggregation layer can further be extended with plausibility or consistency checks (e.g., range validation or more complex logic) to detect implausible or contradictory inputs before aggregation.

Amendments or corrections do not overwrite previous values; instead, new records are appended, ensuring that updates remain fully traceable while the historical state is preserved. This pattern naturally generalizes to multi-organization supply chains. Each supplier can maintain its own machine-level contracts and aggregation contract, forming a structured tree of verifiable records. Higher-level aggregation contracts can then combine data across suppliers when a product moves along the supply chain. This allows multiple suppliers to contribute authenticated data fragments that, when combined, can form a DPP.

3.4. DPP with verifiable claims

The output of this pipeline is a dataset with verifiable claims about the production process, backed by cryptographic proofs from the underlying blockchain layer. The dataset can then be linked with the corresponding DPP, enabling trusted data provenance and compliance verification to the end-user. A key advantage of this approach is that each claim can be traced back to its original emitting source, whether a specific machine, process, or facility. In future deployments that include confidential attributes, the same architecture can be extended without structural changes. Sensitive fields can remain off-chain with integrity hashes recorded on-chain, or be committed on-chain in encrypted form. In both cases, the blockchain layer continues to guarantee data integrity and traceability.

4. Implementation

The architecture described in Section 3 is implemented using the Python programming language. To maintain modularity and flexibility, both the LLM and blockchain layer are designed as separate classes in an object-oriented manner. Fig. 2 illustrates the UML diagram, which outlines the attributes and methods for each class.

4.1. LLM mapping

Starting with the LLM Mapping class, it implements the core logic for transforming unstructured or semi-structured machine operation data into normalized JSON objects. In accordance with object-oriented principles, the class encapsulates functionality for prompt generation, LLM invocation, and structured parsing, supporting various prompting strategies, model backends, and output validation requirements.

The constructor provides a set of configuration parameters, including the LLM provider, model name, prompt type, and structured output method. During initialization, these parameters are stored as internal attributes to enable the instantiation of distinct mapping agents for different machines or data types. Based on the provided parameters, the class initializes the corresponding LLM client and prompts for later execution. This process utilizes the LangChain library (Chase, 2022), which includes configurable clients for various LLM providers and a set of tools to generate prompts via templates dynamically.

The current implementation supports OpenAI and Ollama as LLM providers to demonstrate compatibility across proprietary services and locally running LLMs, while also allowing for the further integration of all available providers. Depending on the selected strategy, LangChain's structured output integration is used to instruct the LLM to return responses that conform to a specific schema or pattern. By selecting "JSON Mode", the LLM is instructed to return well-formed JSON only. With "JSON Schema", on the other hand, the LLM is forced to output instances of a predefined JSON schema constantly.

The prompt template is dynamically assembled using external factory functions that return strategy-specific prompts, each containing a dedicated slot for the raw input JSON. From a meta perspective, two prompting approaches can be distinguished for solving the JSON mapping problem: **Direct Mapping**, a declarative approach where the LLM is prompted to return a standardized object directly for each input instance; And **Mapping Function**, a programmatic approach where the LLM generates a Python function that performs the transformation procedurally. In a production setting, the latter approach is efficient when a machine's data format is stable, as the LLM generates the transformation logic once, which can be reused without repeated inference, thereby reducing cost and latency. However, when the generated mapping function is closely tied to the specific field names and structure of the input sample, it may lack adaptability to semantic and syntactic variations in the input data format. In turn, this could require regeneration or manual adjustment of the function. In such cases, the direct mapping strategy excels by allowing the LLM to react to such variations for each separate input. The proposed implementation supports both techniques via dedicated class methods.

During execution, the selected LLM is invoked using the populated prompt template via LangChain's pipeline mechanism. This setup ensures consistent prompt formatting and model interaction across model providers. The raw response is then parsed, either as a structured JSON object (Direct Mapping) or as a Python code snippet (Mapping Function), which is compiled and called immediately to obtain the transformed output. In both cases, the output is validated using the Pydantic library (Colvin et al., 2025) and a predefined schema to ensure conformance with the expected format. Robust error handling captures issues such as internal LLM exceptions, invalid JSON, or schema mismatches, and classifies them for downstream diagnostics and analysis. Notably, careful error handling and logging are essential to ensure the traceability of input data and transparency of failures, thereby preventing silent errors that could result in incomplete or inconsistent records on the blockchain.

4.2. Blockchain integration

This section describes how the blockchain layer transforms normalized sensor data into verifiable, cumulative claims. In the presented prototype, this blockchain layer is implemented using the Waves blockchain,[3] and all SC-related logic is written in the Ride programming language. Waves was selected primarily for pragmatic reasons, including a mature development toolchain, accessible APIs for SC interaction, and the authors' prior familiarity with the platform, which facilitated rapid prototyping. Importantly, the architecture is intentionally designed in a modular manner, allowing the underlying distributed ledger technology to be replaced without affecting the upstream data-mapping pipeline or the overall system logic.

Each production machine has a dedicated SC, and a separate contract handles daily aggregation. Waves is well-suited to this pattern because each address maintains a versioned key–value store in which all updates are recorded on-chain. This allows access to historical changes and supports the time-series nature of the production metrics. The Ride programming language executes deterministically, which means that range checks and other plausibility rules run on-chain without requiring an external oracle.

Each machine's SC receives a canonical JSON string that conforms to the target schema, as ensured by the LLM mapping stage outlined in Section 3.2. The contract first parses the payload and verifies that the expected fields are present and fall within trivial boundaries (e.g., operating hours must not exceed a day). After validation, the contract writes individual metrics to composite keys that combine the machine address and date, and updates cumulative totals, such as overall water consumption. Because of the nature of the Waves data store, the comprehensive, field-level history for each machine is accessible on-chain, ensuring highly granular traceability. Furthermore, the SCs update cumulative totals (e.g., overall water consumption) in dedicated keys, enabling evaluation of longitudinal consistency directly on the ledger.

A separate contract consolidates the machine data at the line or batch level for a specified date. It reads the relevant keys from each machine contract, aggregates the values to produce totals, and commits them to its data store. While not included in the current implementation, the contract could run additional cross-metric sanity checks at runtime. For example, the SCs could confirm that reported material waste never exceeds the corresponding material input. If any test fails, the transaction is rejected, and a descriptive error flag is logged, preventing illogical data from being stored in the SCs' data storage.

On the client side, the WavesConnector class, depicted in Fig. 2 on the right, signs and submits the canonical JSON to the corresponding machine SC and triggers aggregation at the end of each shift. The WavesConnector is a wrapper around the Waves Python library and implements the abstract BlockchainConnector class, which enables flexible replacement of the blockchain module. Technically, the class hierarchy can be extended to also support non-blockchain storage solutions, but the current implementation focuses on blockchain-based anchoring and aggregation. Once the concrete DPP specifications are finalized, these anchors can be referenced directly to link the DPP to verified and traceable claims. The finalized passport could then be published to the enterprise data space behind the product's QR code, providing stakeholders with cryptographically verifiable access to the production footprint.

While the prototype implementation stores data in plaintext for demonstration and evaluation purposes, the architecture is designed to support the protection of confidential product or supply-chain information in deployment scenarios. Owing to its modular blockchain design, the system allows sensitive information to be handled flexibly: restricted attributes can either remain off-chain with their integrity ensured via on-chain references, or be stored directly in SC data storage in an encrypted format with controlled access. In such cases, only stakeholders who possess the corresponding decryption key can access the underlying values, while the blockchain continues to provide tamper-evidence and traceability.

[3] The official documentation of the Waves blockchain is available at https://waves.tech/

Waves SCs are space-efficient, with each metric entry requiring minimal storage, enabling the system to scale effectively. When a new metric is introduced, only the schema included in the LLM prompt and the contract's validation logic need to be updated. Historical queries return zero for dates before the metric's existence, maintaining semantic correctness without requiring data migration.

In the prototype, the transactions are signed by a key managed within the blockchain connector service, which runs off-chain. This setup is sufficient for experimentation and evaluation. In production deployments, the same architecture can integrate Hardware Security Modules or secure key vaults to protect private keys and ensure that signing keys never leave controlled environments. In such cases, the SCs enforce write permissions by accepting state updates only from the public keys registered in their access-control tables. These access-control entries can be updated through a privileged call, allowing write access to be reassigned when machines are replaced or when another supplier or service provider joins the ecosystem.

5. Evaluation

This section evaluates the proposed pipeline in terms of accuracy, speed, and robustness in a production-like environment. The goal is to understand how well different LLMs perform in mapping unstructured machine data into a hypothetical DPP-aligned format and how the system behaves under varying prompt strategies and complexity levels. The evaluation examines both the correctness of the data mappings and the practical performance of blockchain-based storage under realistic workloads.

5.1. Production scenario

To emulate a real-world production environment, the evaluation builds on the production line introduced in Section 3.1. Specifically, the considered scenario involves a product that requires a DPP at the *batch level* granularity and 10 machines operating sequentially on the production floor. With the production line completing one batch per day, the operational data from each machine is collected and aggregated daily. To enable an autonomous and tamper-evident DPP construction, the prototype addresses three key challenges: (1) Mapping the heterogeneous daily machine data into a standardized JSON format using the LLM-based mapping mechanism, (2) storing the normalized data on the Waves blockchain via SCs assigned to each machine, and (3) aggregating the machine-level data at the end of each batch cycle to form traceable and verifiable claims referenced in the final DPP.

5.2. Method

To evaluate the system within the described production scenario, a comprehensive experimental setup was designed, which enables benchmarking accuracy and speed across multiple configurations, while maintaining identical datasets and hardware conditions. Each run configuration was evaluated using a dedicated script that aggregates key performance metrics, including per-machine statistics about speed, accuracy, and error types.

Mapping accuracy was calculated using a strict exact-match criterion, under which a mapping is considered correct only if all required target fields are present and exactly match the ground truth. This all-or-nothing definition reflects DPP requirements, under which incomplete mappings are unusable. Any missing, incorrect, or hallucinated field renders the entire mapping incorrect. While prior work on LLM-based semantic mapping often reports precision and recall (Section 2.2), evaluating complete JSON instances in this way makes exact-match accuracy the most appropriate and transparent metric for this study.

The comparison of mapping accuracy was conducted across three primary dimensions: **Language models**, ranging from proprietary cloud solutions to locally hosted variants served via Ollama; **Prompting strategies**, including direct mapping through zero-shot and few-shot prompts, as well as the generation of explicit mapping functions; And **mapping tasks**, where the LLMs were exposed to varying levels of complexity and problem types. To support the latter, a novel dataset consisting of source–target pairs is introduced in Section 5.3.1.

Four language models were used to benchmark mapping performance across different capabilities and deployment scenarios. GPT-4o (OpenAI, 2023) was selected as a proprietary baseline representing state-of-the-art performance. Gemma 3 (12B) (Gemma Team, 2025) served as a general-purpose, locally deployable medium-sized model. Codestral (22B) (Jiang et al., 2023) was included to assess the impact of code-specific fine-tuning on the mapping task. Finally, Deepseek-R1 (14B) (DeepSeek-AI, 2025) was chosen for its efficient reasoning capabilities, enabling direct comparison between regular and thinking models.

Following the core accuracy evaluation, which is carried out in Section 5.5, the analysis was extended to compare GPT-4o (proprietary) with locally hosted models, assess differences between reasoning-oriented and general-purpose local models, and evaluate the effectiveness of direct JSON mapping versus mapping function generation. These settings were selected to explore trade-offs relevant to production deployments and constrained environments, with a focus on performance, deployability, and prompt strategy robustness. At this stage, statistical analysis was applied to determine the significance of performance differences across configurations.

The runtime of each run was measured for both the LLM-based mapping, which is mainly determined by model inference time, and the blockchain component, which includes SC calls and data propagation. Since blockchain performance does not depend on the LLM or prompting strategy, it was measured only for the best-performing configuration from the accuracy evaluation. Execution time was recorded by placing a timer at the start and end of each pipeline iteration. The elapsed time was recorded for each sample, accumulated across runs, and then averaged to produce the final performance metrics. Results of the speed comparison are presented in Section 5.5.4.

Beyond the three main evaluation dimensions, the pipeline also allows testing additional settings, such as including the schema in the prompt, forcing the model to follow a fixed output format, and allowing it to retry after making an error. The performance impact of these additional parameters was explored through an ablation study, presented in Section 5.6, which breaks down the individual contributions to the mapping accuracy.

5.3. Dataset and pipeline

Synthetic datasets and an evaluation pipeline were constructed to assess the system's ability to map raw machine data to standardized formats using LLMs. The pipeline natively includes data storage and aggregation on a blockchain through SCs, following the implementation described in Section 4.2.

The creation of a custom dataset was motivated by a notable gap in the field, where mapping datasets suitable for DPP-related tasks are currently lacking. It also ensured a controlled environment for evaluating system behavior and the quality of transformation.

5.3.1. Dataset generation and structure

The dataset was designed to reflect the production scenario outlined in Section 5.1, where a product batch is completed each day and machine-level data is aggregated to form daily entries in a batch-level DPP. Therefore, each sample in the dataset represents the operational data of one machine, already aggregated for a specific production day. The selected time frame spans 1 January 2024 through 30 June 2024, covering 182 daily batches, which across 10 machines results in a total

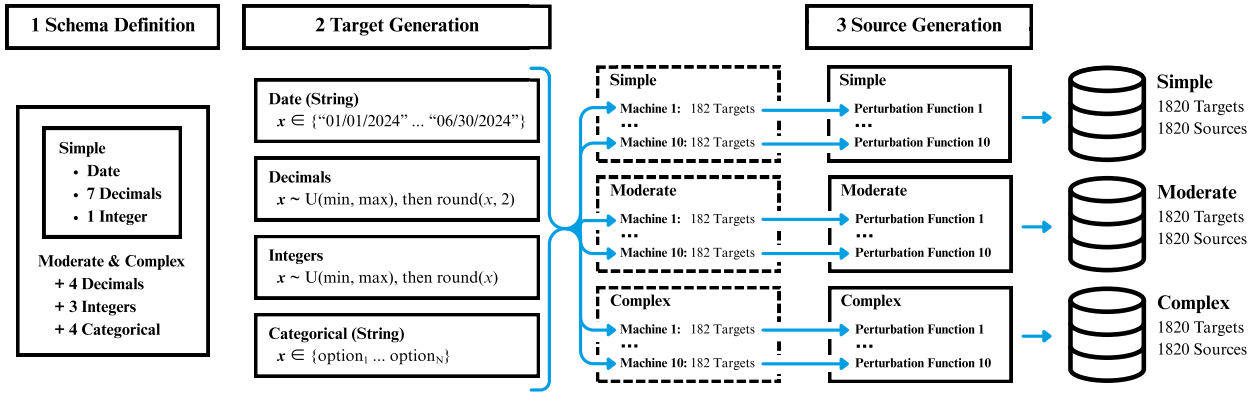

Fig. 3. Dataset Generation Process: Flow diagram visualizing the reverse approach of creating the dataset.

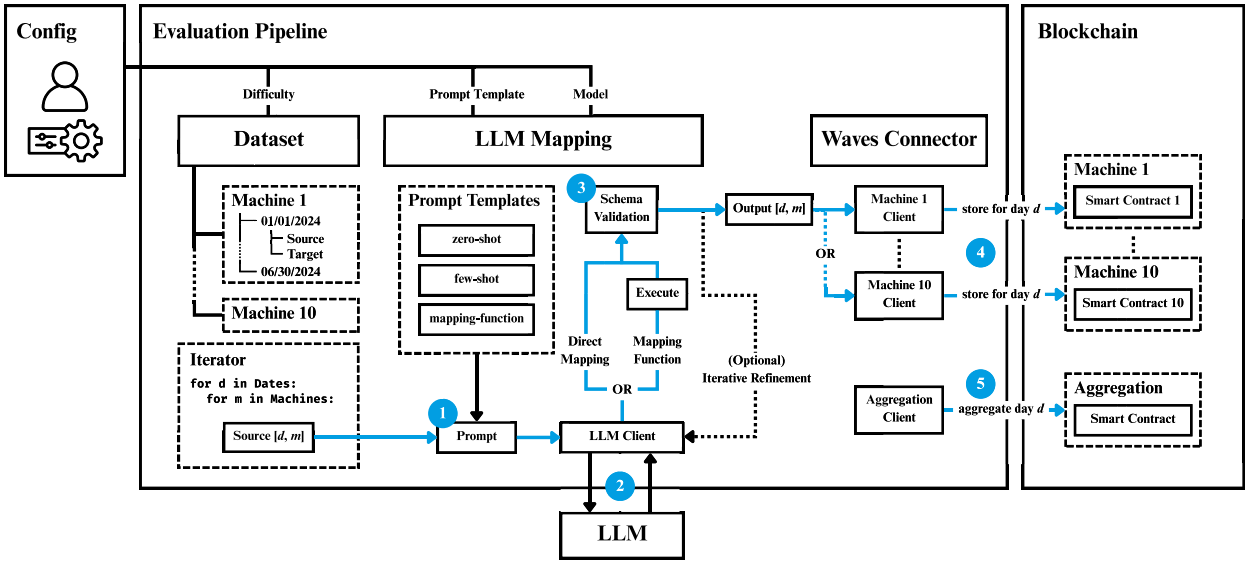

Fig. 4. Evaluation pipeline: A high-level data flow through the pipeline. The blue path and numbered steps illustrate the core process sequence in each iteration. Step 5 is executed every 10 iterations to aggregate the machine data for a specific date.

of 1820 samples. The generation process followed a reverse approach, which is illustrated in Fig. 3.

First, a fixed target schema was designed to simulate the intended DPP structure, as no official DPP schema exists to date. Second, the 1820 target objects were generated by sampling each value from predefined uniform distributions, with numerical values rounded to two decimal places to ensure consistency. Third, for each of the 10 simulated machines, a corresponding perturbation function was defined to transform the target data into a noisy source format (e.g., replacing field names with synonyms or converting units). These perturbation functions apply systematic variations, effectively modeling an ablation of problem types that the LLM needs to resolve. To maintain transparency, summaries of perturbation functions along with representative examples are displayed in Appendix A.1.

Benchmarking under varying levels of complexity was enabled by curating three distinct dataset variants:

- **Simple**: Contains a limited number of numerical attributes per machine, all in well-structured JSON formats. Specifically, each sample includes the date formatted as a string, 7 floats, and 1 integer.
- **Moderate**: Builds on the simple variant, but increases the number of attributes and introduces categorical values, which are modeled as string enumerations and sampled via random choice. Each sample consists of the date, 11 floats, 4 integers, and 4 categorical values.
- **Complex**: Targets share the structure of the moderate set. To increase complexity, the perturbation functions introduce deeply nested objects, multilingual fields, tabular and CSV structures, synthetic machine logs, conflicting or redundant values, and a human worker summary, which embeds the values in plain text.

These complexity levels reflect a progression from clean, synthetic inputs to real challenges in industrial environments. The *simple* variant models minimal, near-ideal machine outputs with clean numerical data. The *moderate* variant reflects more feature-rich environments, where machines report additional parameters and categorical status codes. The *complex* variant emulates real-world challenges such as unstructured logs, deeply nested structures, and human-generated summaries.

Each sample of the resulting dataset comprises a source in an unstructured, machine-specific format and a corresponding target in a standardized, DPP-oriented format, representing the ground truth. Excerpts for each complexity level are provided in Appendix A.2.

5.3.2. *Evaluation pipeline*

The evaluation pipeline, illustrated in Fig. 4, was designed to simulate day-by-day data flow throughout the production line.

To maintain the object-oriented approach described in Section 4, a dedicated class facilitates the dataset generation, local caching, and provides an iterator functionality. The pipeline itself is modeled as a class and instantiates the dataset, LLM mapping, and Waves connector classes based on the provided run configuration.

Each iteration involved (1) populating the prompt template with the current source sample, (2) invoking the LLM through the client, (3) parsing the result and validating the schema, (4) storing a successful mapping on the Waves blockchain by invoking the respective SC, and (5) triggering the aggregation SC once all mappings have been processed for that date (e.g., every 10 iterations for 10 machines). Because *direct mapping* and *mapping function* prompts involve different processing logic, their execution were separated into two distinct methods tailored to each mapping strategy. This separation is visualized in Fig. 4 between steps 2 and 3, where the right path involves an additional step to call the generated Python function before validating the transformed result.

For *direct mapping*, all samples were processed. In contrast, for the *mapping function* runs, the dataset was reduced to a single representative sample per machine, as a successfully generated function can reliably map all subsequent structurally identical samples. Therefore, applying the generated function to all samples was omitted, although it remains technically feasible.

To evaluate the LLMs' capacity for self-correction, an optional refinement loop was implemented that repeatedly invokes the model until validation passes or a defined iteration limit is reached. A configuration parameter controls this functionality, where a value of 0 disables refinement and any positive value sets the maximum number of refinement iterations. Each refinement step enriches the initial prompt with information about previous errors. Notably, this is done without relying on persistent chat history, which keeps the process stateless and leaves room for future extensions involving context tracking or memory mechanisms.

For each iteration within the main loop, mapping results and metadata, including token usage, speed, parsing mistakes, and errors, were logged to a local file for later inspection. Outputs with valid formatting and schema conformance were not rejected even if values were incorrectly mapped (e.g., failed unit conversions or hallucinations), since in real-world production environments, the ground truth is not available for verification.

Once a run was completed, correct samples, error types, token usage, and process speeds were aggregated to calculate the final evaluation statistics. These functions were separated to encapsulate evaluation logic from the primary process.

5.4. Prompts

Prompt templates were used as system prompts to establish a consistent task framing for the LLM. Each prompt was constructed to include an introductory task description, a dynamic middle section specific to the chosen prompt type, and a placeholder for the JSON input passed as the human message. Due to LangChain's object-oriented and flexible design, these templates are model-agnostic and can be reused across different model providers.

The *zero-shot prompt* includes a middle section consisting of a structured schema description generated from the target Pydantic model. This format instructs the model about the required output structure without offering any concrete examples.

In contrast, the *few-shot prompt* populates the middle section with input–output examples that mirror the dataset structure. Four illustrative samples were selected for this purpose, with each one corresponding to a distinct machine in the dataset. They were generated using the same perturbation logic employed for creating the actual dataset, while ensuring that different values were used to prevent the model from recognizing the actual task solution. The examples were generated in advance and cached per complexity level to ensure deterministic behavior and fair comparison across runs.

The *mapping-function prompt* deviates from the others by instructing the LLM to produce a valid Python function given a task description and the target schema. The prompt further includes restrictions concerning the generated code snippet, e.g., that the function must adhere to a specified signature and return type.

To maintain transparency, exact prompt templates used can be found in Appendix B.

Additionally, the pipeline allows for schema inclusion and exclusion in each prompt via a dedicated configuration parameter. This parameter was primarily used for evaluating the necessity of schema presence when structured output is enabled for the model.

5.5. Results

Starting with a comparison across configurations utilizing direct-mapping strategies, Table 1 highlights the highest-performing setup for each model and dataset complexity level based on mapping accuracy.

In addition to the three primary evaluation dimensions (model, dataset difficulty, and prompt type), the table also reports the structured output directive applied for each run. As mentioned before in Section 5.2, accuracy was measured on a sample-level: Since each sample comprises one JSON record, the entire sample was counted as incorrect if any field name or value failed schema validation. Thus, the percentages in the table reflect the share of samples that were transformed flawlessly end-to-end.

Across all twelve entries, the recorded accuracy varied between 30.44% and 99.95%. On complex inputs, Gemma 3 (12B) achieved the lower bound, while GPT-4o reached the upper bound on moderate inputs. The only configurations to surpass the 99% accuracy mark are the two GPT-4o runs, achieved on the moderate and simple datasets. In contrast, the lowest-performing results, all falling below 50%, were observed with Gemma 3 (12B) on both complex and moderate datasets, and Codestral (22B) on complex data.

A clear trend in the results is that accuracy consistently decreases as dataset complexity increases. This pattern is most pronounced in the case of Deepseek-R1 (14B), whose accuracy drops from 93.79% to 43.68%, and Codestral (22B), where it declines from 53.52% to 32.97% over the same range. GPT-4o maintains consistently high accuracy across all difficulty levels, with only a slight improvement as the tasks become simpler, indicating minimal sensitivity to input complexity.

Another observation is that ten out of the twelve configurations displayed in Table 1 use few-shot prompts. The only exceptions are the zero-shot runs that lead on the simple dataset for Codestral (22B) and the complex dataset for Deepseek-R1 (14B). Structured-output directives are also prevalent in the results, with five rows adopting JSON Schema and three using JSON Mode.

5.5.1. GPT-4o versus open-weights models

For each difficulty tier, we compared the distributions of per-record accuracies between GPT-4o and each open-weight model using a two-tailed Mann–Whitney U test. The detailed test results are reported in Table C.7 in Appendix. Shapiro–Wilk tests indicated strong non-normality, confirming that a non-parametric approach was appropriate.

Across the complex and moderate datasets, GPT-4o outperformed all open-weight models with statistically significant differences ($p < 0.05$). For instance, in the complex tier, the comparisons between DeepSeek and GPT-4o as well as between Gemma and GPT-4o yielded $p = 0.005$ and $p = 0.001$, respectively, indicating a clear advantage for GPT-4o. Similarly, in the moderate tier, the comparison DeepSeek vs. GPT-4o resulted in $p < 0.001$.

In the simple tier, however, the difference between DeepSeek and GPT-4o did not reach statistical significance ($p = 0.096$), although the median accuracy of GPT-4o remained higher. This suggests that while

Table 1
Best *direct-mapping* configurations per model and difficulty level with corresponding accuracy.

Model	Difficulty	Prompt	Structured output	Accuracy
Codestral (22B)	simple	zero-shot	none	51.54%
Codestral (22B)	moderate	few-shot	json-schema	53.52%
Codestral (22B)	complex	few-shot	json-schema	32.97%
Deepseek-R1 (14B)	simple	few-shot	json-schema	93.79%
Deepseek-R1 (14B)	moderate	few-shot	none	66.10%
Deepseek-R1 (14B)	complex	zero-shot	none	43.68%
Gemma 3 (12B)	simple	few-shot	json-mode	57.91%
Gemma 3 (12B)	moderate	few-shot	json-mode	50.05%
Gemma 3 (12B)	complex	few-shot	json-mode	30.44%
GPT-4o	simple	few-shot	none	99.89%
GPT-4o	moderate	few-shot	json-schema	99.95%
GPT-4o	complex	few-shot	json-schema	93.13%

Table 2
Best *mapping-function* configurations per model and difficulty level with corresponding accuracy.

Model	Difficulty	Refinements	Accuracy
Codestral (22B)	simple	0	60.00%
Codestral (22B)	moderate	3	40.00%
Codestral (22B)	complex	0	10.00%
Deepseek-R1 (14B)	simple	3	30.00%
Deepseek-R1 (14B)	moderate	3	30.00%
Deepseek-R1 (14B)	complex	3	10.00%
Gemma 3 (12B)	simple	3	60.00%
Gemma 3 (12B)	moderate	0	40.00%
Gemma 3 (12B)	complex	0	20.00%
GPT-4o	simple	0	60.00%
GPT-4o	moderate	0	60.00%
GPT-4o	complex	0	40.00%

GPT-4o consistently achieved the best results, the performance gap in the simplest dataset was smaller.

Overall, these results confirm the descriptive advantage already visible in Table 1, where GPT-4o achieved 99.89%, 99.95%, and 93.13% accuracy in the simple, moderate, and complex tiers, respectively.

5.5.2. Reasoning versus non-reasoning open-weight models

A closer look at the open-weight group shows Deepseek-R1 (14B) leading across all difficulties numerically: With 93.79% (simple), 66.10% (moderate), and 43.68% (complex), the model outperforms Codestral (22B) at 51.54%, 53.52%, 32.97%, and Gemma 3 (12B) at 57.91%, 50.05%, 30.44%. Despite this indication, none of the associated Mann–Whitney–U tests (Table C.7) crosses the significance boundary: In the simple tier, the comparisons between Deepseek-R1 (14B) and both Codestral (22B) and Gemma 3 (12B) result in $p \approx 0.27$. For the moderate tier, both comparisons yield $p \approx 0.85$ In the complex tier, the comparison between Deepseek-R1 (14B) and Codestral (22B) yields $p = 0.54$, while the comparison with Gemma 3 (12B) results in $p = 0.18$". The reasoning-oriented architecture therefore delivers higher median accuracies, but, given the present sample size and variance, not by a margin large enough to be declared statistically reliable.

5.5.3. Direct mapping versus generated mapping functions

To examine the impact of using the mapping function approach instead of direct sample transformation, Table 2 presents the accuracies achieved by the best-performing configurations for each model and dataset difficulty combination.

As noted in Section 5.2, the sample size for this approach was limited to one representative example per machine, based on the assumption that an operational mapping function could generalize to all remaining samples. Consequently, each run included a total of 10 samples, resulting in discrete accuracy values. Despite the additional step of compiling and executing the generated transformation function, all other settings mirrored the direct-mapping setup.

In cases where multiple configurations achieved equal results, the one with the fewest refinement attempts was retained. Thus, the table already reflects the most favorable outcome attainable with code generation, and all comparisons below use those maxima to compare against the direct mapping scores in Table 1.

For Codestral (22B), the generated functions outperform direct mapping on simple inputs, achieving 60% accuracy compared to 51.54%. However, performance declines with increasing complexity: On moderate inputs, generated functions lag by approximately 13% despite three retries. On complex inputs, the gap widens to around 23%.

In the case of Deepseek-R1 (14B), generated functions achieved only 30% accuracy on simple inputs, compared to 93.79% with direct mapping, resulting in a performance gap exceeding sixty percentage points. On moderate inputs, accuracy remained at 30%, trailing direct mapping by over thirty percent. In the complex scenario, even with three retries, function accuracy dropped to just 10%, further widening the performance deficit.

For Gemma 3 (12B), the two strategies converge on simple inputs, with generated functions achieving 60% accuracy compared to 57.91% from direct mapping, a negligible advantage in favor of the function-based approach. In the case of moderate and complex inputs, however, even with three retries, the function mapping variant consistently trails by approximately ten percentage points relative to direct mapping.

While the direct mapping technique remains between 93% and 99% for GPT-4o, the mapping function alternative tops out at 60% for moderate and simple inputs and further decreases to 40% on the complex dataset. Additional retries did not improve these results, leaving the best configurations presented in Table 2 with zero refinement attempts. Inspection of the generated mapping functions revealed a consistent pattern: The LLMs produced static functions that directly return a dictionary with hard-coded source-to-target mappings. This behavior aligns with the assumption outlined in Section 4.1, indicating a reliance on a fixed input format. More flexible techniques, such as fuzzy matching or if-else logic, could not be recorded in the model responses. Consequently, the generated functions failed to execute if a key did not match the input schema.

Overall, the results of this comparison reveal a clear trend: Models frequently fail to produce valid Python functions, making function-based mapping consistently less effective than direct mapping, with only a few isolated exceptions.

Table 3
Comparison of LLM inference times (in seconds) per sample across different models.

Model	Avg (s)	Std (s)	Min (s)	Max (s)	P_{95} [a] (s)
Codestral (22B)	11.77	6.05	0.87	106.14	21.69
Deepseek-R1 (14B)	18.29	79.30	3.19	13 944.11	62.05
Gemma 3 (12B)	6.53	2.95	2.77	63.03	9.36
GPT-4o	2.52	1.63	0.90	201.09	4.30

[a] P_{95}: 95th percentile — the value below which 95% of the measurements fall.

Table 4
Ablation Study: Mapping accuracies of the evaluated models across different prompt and output configurations, measured exclusively on the complex dataset variant. * RA = Refinement Attempts.

Prompt	Schema included	Structured output	RA*	GPT-4o	Gemma 3 (12B)	Codestral (22B)	Deepseek-R1 (14B)
zero-shot	✓	×	0	84.51%	25.44%	26.76%	43.68%
zero-shot	✓	json-mode	0	85.49%	25.27%	28.41%	–
zero-shot	✓	json-mode	3	–	25.27%	28.46%	–
zero-shot	×	json-schema	0	79.67%	24.67%	24.45%	31.10%
zero-shot	✓	json-schema	0	81.43%	25.27%	28.25%	33.63%
few-shot	Enums	×	0	93.02%	29.62%	20.93%	28.96%
few-shot	Enums	json-mode	0	92.69%	29.62%	31.32%	–
few-shot	Enums	json-mode	3	–	30.44%	32.09%	–
few-shot	×	json-schema	0	93.02%	29.45%	30.82%	42.03%
few-shot	Enums	json-schema	0	93.13%	29.12%	32.97%	42.09%
mapping-function	✓	×	0	40.00%	20.00%	10.00%	0.00%
mapping-function	✓	×	3	–	20.00%	10.00%	10.00%

5.5.4. Execution speed

The runtime performance of the overall system was measured, encompassing the time required to transform raw machine data into a standardized format and store it on the blockchain. This aims to evaluate whether the architecture complies with the practical constraints necessary for real-world deployments in industrial settings.

A dedicated full-stack run was conducted using the GPT-4o model, during which all 1820 machine-level records were successfully written to the Waves blockchain. Despite the overhead typically associated with decentralized storage, the system achieved an average time of 3.77 s per transaction for complete propagation. The observed standard deviation was 1.10 s, with minimum and maximum times recorded at 163 ms and 7.25 s, respectively.

The blockchain component was excluded from the remaining runs, as storage times are independent of the LLM-based mapping system under evaluation. To facilitate comparison between models, timing statistics for the best-performing configurations are presented in Table 3.

All execution times were measured on the same machine to ensure consistency, with local models served via Ollama on a single NVIDIA Quadro RTX 8000 GPU. The GPT-4o model exhibited the fastest mapping performance overall, requiring an average of 2.52 ± 1.63 s per mapping. Despite being generally slower due to the given hardware specifications, local models still performed within practical ranges: Gemma 3 (12B) averaged 6.53 ± 2.95 s (min: 2.77 s, max: 63.03 s), Codestral (22B) took 11.77 ± 6.05 s (min: 0.87 s, max: 106.14 s), and Deepseek-R1 (14B) had the widest spread with 18.29 ± 79.30 s (min: 3.19 s, max: 13,944.11 s). The maximum value for Deepseek-R1 (14B) occurred only once and was caused by the model entering an infinite loop, repeatedly generating the same reasoning sentence until the token limit of 320,000 was reached. The 95th percentile for Deepseek-R1 (14B) is 62.05 s, still higher than the others, but within a much more reasonable range.

To rigorously evaluate the differences in execution time across models, a statistical analysis was conducted on the LLM inference durations presented in Table 3. First, the distributions were tested for normality using the Shapiro–Wilk test, which consistently rejected the null hypothesis of normality ($p < 0.05$) across all models and metrics, indicating skewed timing distributions. These test results are reported in Table C.8 in Appendix. As a result, all pairwise comparisons were performed using the Mann–Whitney U test (Table C.7).

To go beyond statistical significance and assess the practical relevance of observed differences, the analysis was further extended to include Cliff's Delta as an effect size measure. This non-parametric statistic quantifies how often values from one distribution are higher than those from another. The results showed that GPT-4o significantly outperformed all local models with $p < 0.0001$ and Cliff's Delta values exceeding 0.9, indicating a strong and consistent performance advantage in terms of latency.

For instance, compared to Codestral (22B), GPT-4o was not only statistically faster ($p < 10^{-6}$), but also practically dominant with $\delta = 0.99$, denoting that nearly every timing measurement from GPT-4o was faster than the corresponding value from Codestral (22B). A similar pattern was observed when comparing GPT-4o to Gemma 3 (12B) and Deepseek-R1 (14B), both of which recorded large effect sizes ($\delta = 0.85$ and $\delta = 0.98$, respectively).

Gemma 3 (12B) outperformed Codestral (22B) with a large effect size ($\delta = 0.53$), indicating a clear practical latency advantage. The same was observed in total execution time with $\delta = 0.52$, further validating that Gemma 3 (12B) yields stronger performance in this pairwise comparison.

In contrast, although the difference between Codestral (22B) and Deepseek-R1 (14B) was statistically significant ($p < 0.000001$), the Cliff's Delta was only $\delta = 0.13$, reflecting a negligible practical difference. This suggests that while Deepseek-R1 (14B) had slightly lower median latency in that specific comparison, the effect was inconsistent across samples and likely not meaningful in practice.

These findings underscore that GPT-4o is not only statistically but also practically superior in terms of latency, making it more suitable for latency-sensitive industrial settings. In contrast, the open-weight models remain viable for non-critical throughput scenarios but exhibit latency risks and occasional instability, as exemplified by the infinite loop incident observed with Deepseek-R1 (14B). Further research is required to mitigate this issue, e.g., through the implementation of additional safeguards or capped inference durations.

5.6. Ablation study

An ablation study was conducted to evaluate the individual effects of structured output methods, iterative refinement, and in-/excluding the target schema in the prompt. Table 4 provides a detailed summary of the configurations and their corresponding mapping accuracies

Table 5
Token Usage: Average input and output tokens per request across different prompt and output configurations, measured exclusively on the complex dataset variant and without iterative refinement. ↓ = input tokens, ↑ = output tokens.

Prompt	Schema included	Structured output	GPT-4o	Gemma 3 (12B)	Codestral (22B)	Deepseek-R1 (14B)
zero-shot	✓	×	↓ 1,411 ↑ 230	↓ 1,563 ↑ 273	↓ 1,725 ↑ 480	↓ 1,502 ↑ 1,316
zero-shot	×	json-schema	↓ 1,094 ↑ 166	↓ 550 ↑ 268	↓ 580 ↑ 338	↓ 519 ↑ 269
zero-shot	✓	json-schema	↓ 2,065 ↑ 166	↓ 1,564 ↑ 261	↓ 1,725 ↑ 322	↓ 1,502 ↑ 255
few-shot	Enums	×	↓ 2,900 ↑ 203	↓ 3,768 ↑ 265	↓ 4,009 ↑ 349	↓ 3,557 ↑ 1,177
few-shot	×	json-schema	↓ 3,352 ↑ 165	↓ 3,700 ↑ 256	↓ 3,927 ↑ 310	↓ 3,491 ↑ 247
few-shot	Enums	json-schema	↓ 3,417 ↑ 165	↓ 3,768 ↑ 256	↓ 4,009 ↑ 309	↓ 3,557 ↑ 252
mapping-function	✓	×	↓ 1,455 ↑ 481	↓ 1,615 ↑ 771	↓ 1,775 ↑ 667	↓ 1,545 ↑ 2,257

across all models. All results are based exclusively on the complex dataset variant, as it incorporates the highest diversity in mapping tasks.

For GPT-4o, runs involving the iterative refinement component were intentionally omitted to reduce API usage costs. The three missing values for Deepseek-R1 (14B) are due to the unavailability of the JSON mode structured output feature in the Ollama interface at the time of experimentation. As shown in the second column of Table 4, only the enumerated values were included in the few-shot prompts. This design assumes that the provided examples offer sufficient guidance on the target schema, while also requiring the model to internalize the complete set of possible categorical values.

Table 4 shows that few-shot prompting yields the highest mapping accuracies across all models, except for the first row, where Codestral (22B) performs better using the zero-shot approach. Adding the target schema to the prompt resulted in slightly better performance across all models. For runs where the schema was omitted from the prompt, the model relied solely on guided decoding through structured outputs to enforce the schema. Similar to the observations in the previous sections, iterative refinement yielded only negligible improvements, and the mapping-function baseline performs poorly, underscoring the need for dynamic mapping capabilities in LLMs.

In the following two sections, emphasis was placed on assessing whether enabling the plain JSON mode or the more advanced JSON schema mode leads to statistically significant improvements in mapping accuracy and how the types of errors shift across configurations.

5.6.1. Impact on mapping accuracy

Although LLM providers typically recommend including the schema description in the prompt when using structured outputs, the results show only a minor performance gain from doing so. Enabling structured output directives did not lead to statistically significant improvements in mapping performance. Mann–Whitney–U tests were conducted between runs using no structured output (×) and those using either plain JSON mode or JSON schema, grouped by model and averaged over all difficulty levels (Table C.7).

The comparison between unstructured and structured configurations yielded a p-value of 0.67 for Codestral (22B) and 0.92 for Gemma 3 (12B), providing no statistical evidence to reject the null hypothesis that both configurations perform equivalently. For Deepseek-R1 (14B) and GPT-4o there was also no significant difference. This result indicates that structured output enforcement, at least in isolation, is not a statistically reliable mechanism for increasing the rate of successful mappings.

5.6.2. Error type distribution

Although mapping accuracy remained largely stable, error type distributions shifted markedly. For instance, Codestral (22B) produced invalid JSON outputs at rates of 52.86% in the zero-shot and 13.9% in the few-shot unstructured settings. Further inspection of the raw model responses revealed a consistent error pattern: the model attempted to output the calculation for unit conversion (e.g., dividing by a factor of 1000 for *ml* to *l*), which, despite being mathematically correct, resulted in syntax errors. After applying the plain JSON mode, the number of invalid JSON exceptions diminished entirely. This behavior is expected, as these modes enforce a JSON-compliant response through a guided decoding process. As such, similar results were observed for the remaining models, where structural failures were eliminated when JSON output was enforced.

Unfortunately, approximately 20% of the JSON records generated by Codestral (22B) were syntactically valid but failed to comply with the predefined JSON schema, leading to Pydantic validation errors. Through the help of iterative refinements, these errors could be reduced to almost zero, but without improving the overall accuracy. Once the strict JSON schema mode was applied, which restricts generated tokens to those permitted by the schema, all validation errors were eliminated.

However, these improvements in syntactic validity came at the cost of semantic fidelity. The false positive rate, referring to correctly formatted records containing incorrect values, nearly doubled for both Codestral (22B) and Gemma 3 (12B) when JSON schema enforcement was enabled. At the same time, the other models also showed a noticeable, though more minor, increase. These findings imply that structured prompts ensure format compliance but may also encourage the model to produce seemingly correct outputs that are semantically misaligned with the ground truth.

5.6.3. Token usage

In LLMs, text is first tokenized into smaller units such as words or subwords to enable sequential processing. Since tokenization strategies differ between models, the number of tokens representing the same text can vary significantly. In this ablation study, token consumption was measured separately for input tokens (provided to the model) and output tokens (generated by the model). This distinction is especially relevant for proprietary models and the economic factor, as output tokens are usually more expensive. Table 5 displays the average input and output tokens per sample from the *complex* difficulty tier, collected during the experiments of the previous ablation study.

Table 5 reveals several consistent trends across models and prompting strategies. Input tokens dominate the total count, as they include the target schema and few-shot examples in the respective configurations, while the output is restricted to the structured JSON. As

explained before, token counts differ across models due to tokenizer design. In most cases GPT-4o shows the lowest overall token consumption, followed by Gemma 3 (12B), Codestral (22B), and Deepseek-R1 (14B). The latter consumes substantially more tokens when reasoning is enabled, but fewer when structured output is enabled, indicating that its reasoning process is the primary contributor to token inflation rather than tokenizer inefficiency. Including few-shot examples and schema information increases token usage by a factor of two to three compared to zero-shot configurations. However, this increase is offset by improved mapping accuracy, as detailed in Section 5.6.1.

An anomaly occurs in GPT-4o token accounting: Enabling the structured output directive increases input token counts even though its mere activation should not alter prompt size. This suggests that OpenAI's API processes the schema via an implicit system-level prompt, unlike local inference frameworks such as Ollama, where token counts remain constant.

For Deepseek-R1 (14B) configurations without structured output, token usage inflates by 3–4× relative to other models. Subject to the internal implementation of Ollama, enabling structured output disables explicit reasoning and thus reduces tokens, but also degrades zero-shot accuracy by $\approx$ 10% as shown in Table 4. Conversely, in few-shot settings, accuracy improves by $\approx$ 12% when structured output is enabled. This indicates that DeepSeek's internal reasoning is most beneficial when task instructions are extensive and self-contained (zero-shot), while few-shot examples provide sufficient guidance even without reasoning. Attempts to preserve reasoning through schema nesting (e.g., using a dedicated "thinking" attribute) did not improve results and occasionally degraded output quality.

5.7. Discussion

Although the mapping accuracy was observed to decrease significantly for local LLMs as complexity increased, the evaluation results validate the practicality of the proposed system architecture for real-world use, particularly for daily batch-level construction of DPPs. The pipeline demonstrates reliable end-to-end operation (especially for GPT-4o runs), combining LLM-based data transformation with blockchain-backed storage and achieving acceptable performance across all core components.

From a system latency standpoint, the architecture performs robustly even under high load. Using the GPT-4o model, the prototype successfully processed and stored all 1820 machine-level records on the blockchain, achieving an average write time of 3.77 s per transaction. These results confirm that the blockchain integration layer introduces only a minor and well-contained delay, which remains fully compatible with typical daily production cycles where immediate post-manufacturing data availability is rarely necessary.

In terms of mapping speed, GPT-4o leads all models with an average latency of 2.52 ± 1.63 s per transformation. Open-weight alternatives, such as Gemma 3 (12B), Codestral (22B), and Deepseek-R1 (14B), exhibited slower average run times during execution on a single GPU, despite being optimized for inference through the Ollama software. A substantial variance in generation time was observed for Deepseek-R1 (14B), with occasional outliers exceeding $13,000$ s, attributed to excessive reasoning steps and instances of the model entering infinite generation loops. This highlights potential instability, which is particularly concerning for unattended or time-sensitive deployments.

More critical than runtime, however, is the quality of the produced mappings. The evaluation revealed a stark performance gap between the proprietary GPT-4o and the local models. On the proposed benchmark dataset, GPT-4o achieved near-perfect accuracy by effectively handling deeply nested objects, multilingual inputs, and other comprehensive mapping tasks.

In contrast, the open-weight models delivered much weaker performance. For instance, Gemma 3 (12B) did not exceed 60% accuracy on the simple set and reached only 30.44% on complex inputs. Codestral (22B) and Deepseek-R1 (14B) performed slightly better than other open-weight models. Still, they fell short compared to GPT-4o across all difficulty tiers, indicating that neither specialization in coding tasks nor advanced reasoning capabilities alone can compensate for the overall performance gap. A series of Mann–Whitney U tests confirmed that these differences are not only descriptive but also statistically significant. In every complexity tier, GPT-4o outperformed the others with $p < 0.05$, affirming its reliability for critical mapping tasks where flawless transformations are required.

Among the open-weight models, Deepseek-R1 (14B) outperformed Codestral (22B) and Gemma 3 (12B) in accuracy, particularly in simple and moderate complexities. None of the observed pairwise differences within the open-weight models reached statistical significance, suggesting that the variance remains too high or the effect sizes too small to support definitive conclusions. Nevertheless, the descriptive pattern suggests that Deepseek's reasoning-oriented architecture offers some advantages in lower-complexity settings.

A more unexpected outcome emerged from the mapping-function prompt experiments, where LLMs were tasked with generating executable Python functions to transform the input data. While the approach enhances interpretability and reusability and effectively leverages the LLMs' strong code generation capabilities, the resulting functions primarily rely on static, hard-coded mappings. Across all models, the mapping-function technique consistently underperformed compared to direct mapping. For instance, GPT-4o dropped from 99.95% (direct mapping) to 60.00% (function mapping) on moderate inputs, and from 93.13% to 40.00% on complex ones. Even when enabling up to three refinement attempts, the best configurations could not close the gap. This suggests that current LLMs, even at state-of-the-art capacity, struggle to generate robust and adaptable transformation code when provided solely with the target schema. More advanced prompting strategies, such as incorporating explicit instructions for handling flexible schemas, could enhance the adaptability and robustness of the generated functions.

A second noteworthy observation was that using structured outputs did not lead to a significant performance improvement. Although it ensures strict adherence to the expected schema, it was often accompanied by an increase in incorrect values, particularly in the case of Codestral (22B). One possible explanation is the critical intervention into the LLM's token sampling process, which may force token selections that deviate from the model's intended output. This disruption can interfere with the model's generative flow and compromise its ability to engage in coherent, self-reflective autoregressive reasoning. In turn, the strict format constraints enable structural compliance, but come at the expense of semantic accuracy, potentially leading to overconfidence in superficially valid but incorrect outputs.

The ablation on token usage clarified that input tokens dominate overall usage, as they include a thorough task description, target schema information, and few-shot examples, where applicable. In contrast, model outputs from non-reasoning models remain concise and schema-bound, as they are constrained by the prompt and structured output directives. Token efficiency varies between models, with GPT-4o showing the lowest consumption and Deepseek-R1 (14B) the highest when reasoning is active, confirming that its reasoning process drives token inflation. Structured output disables the thinking preamble entirely for Ollama-based reasoning models like Deepseek-R1 (14B)and attempts to preserve it through workarounds, but these workarounds failed to restore accuracy. As a result, the unstructured zero-shot configuration remains the most suitable setup for maintaining reasoning quality and overall performance. For the proprietary GPT-4o model, enabling structured output increases the input size, likely due to internal schema processing.

Taken together, these results highlight both the strengths and the current limitations of the proposed pipeline. The architecture itself proves effective: LLM-based transformation and blockchain storage

introduce minimal latency, and the full-stack implementation scales reliably to real-world data volumes. By emphasizing modular implementation, system components can be replaced with alternative solutions or upgraded independently, enabling flexible adaptation to evolving requirements. GPT-4o delivers exceptional accuracy with fast processing speeds, making it highly suitable for industrial applications. Open-weight models, although attractive due to their data-privacy benefits, perform significantly worse in both stability and mapping quality under the current evaluation setup. Additional techniques such as fine-tuning, domain-specific augmentation, or comprehensive prompt engineering may be necessary to close this gap in future iterations.

6. Conclusion

This conclusion revisits the three research questions by summarizing the proposed architecture (RQ_1), the observed performance characteristics under realistic throughput (RQ_2), and the influence of model choice and prompting strategy on mapping accuracy (RQ_3).

6.1. Summary

This work began by outlining the background, motivation, and research questions in Section 1. It then examined the state of the art in Section 2, covering the evolving regulatory landscape for DPPs, the use of LLMs for data harmonization, and the role of blockchain in ensuring traceability and verification within DPP systems. In Section 3, a modular architecture was proposed for generating DPPs from raw sensor data, combining LLM-based standardization with blockchain-backed storage to ensure persistent and verifiable records. The system's implementation was described in Section 4, where Python was used together with the LangChain and Pydantic libraries for prompt management and data validation, respectively. Blockchain persistence was realized through SCs written in Ride, deployed on the Waves blockchain, enabling verifiable and decentralized storage of DPP-related records. To evaluate the system, a comprehensive pipeline was developed in Section 5.3, featuring a novel dataset designed to reflect day-to-day production mapping tasks of varying complexity. Specifically, the pipeline involved heterogeneous sensor payloads from ten machines that were sequentially entered into the system, normalized by the LLM component, validated against a predefined schema, and then written to machine-specific SCs. A dedicated aggregation SC composes these per-machine records into shift-level DPP-ready datasets, yielding an audit trail whose granularity can be tuned by instantiating additional SCs or nesting them hierarchically. In Section 5.5, the performance of LLM-based mapping was examined across multiple models and configurations, supported by a detailed ablation study.

6.2. Findings

The evaluation showed that the modular architecture proposed in Section 3 delivers a practical integration of LLM-based mapping and blockchain persistence. Results in Section 5.5.4 emphasized that this design scales linearly to at least 1 820 records, with a median LLM-mapping time of 2.52 s (GPT-4o) and a median on-chain commit time of 3.77 s. These findings satisfy both $\mathbf{RQ_1}$ (system architecture) and $\mathbf{RQ_2}$ (operational performance) by demonstrating that verifiability, access control, and throughput can be achieved simultaneously through a set of loosely coupled micro-services backed by SCs. Replacing the proprietary GPT-4o with open-weight models such as Gemma 3 (12B), Codestral (22B), and Deepseek-R1 (14B) led to increased mapping times when running on a single GPU, even with inference optimizations provided by Ollama. Deepseek-R1 (14B) further showed a high variance in terms of speed and occasionally entered infinite generation loops, which displays a limiting factor in uncontrolled or time-sensitive environments.

To answer $\mathbf{RQ_3}$, four language models and three prompting approaches were compared across all dataset tiers. GPT-4o in conjunction with a few-shot prompt delivers near-perfect mappings above 99% accuracy on simple and moderate inputs and 93.13% on complex ones (for details, please refer to Table 1). A similar configuration yielded the best performance for the open-weight models, as well. Yet, even the strongest contender, Deepseek-R1 (14B), trails GPT-4o by large margins once mapping tasks involved complex structures or multilingual noise, and Mann–Whitney U tests confirmed the performance gap as statistically significant ($p < 0.05$). Zero-shot prompts are only helpful in isolated, low-complexity cases, and asking LLMs to generate transformation functions consistently yields worse results. For instance, the performance of GPT-4o dropped from 99.95% (direct mapping) to 60% (mapping function) on moderate tasks. Unfortunately, giving LLMs the ability to retry the mapping based on previous errors did not provide a significant performance boost or any benefit at all. An ablation study further revealed that structured-output directives remove syntactic errors for the open models but often intensify semantic drift, whereas GPT-4o remains robust in both cases. These findings establish *few-shot prompting with explicit schema guidance* as the most reliable strategy for high-end models, but also highlight the need for further exploration on how to leverage locally deployed LLMs effectively for JSON mapping tasks.

6.3. Key benefits for businesses

Although the results of this study have not yet been validated in collaboration with industry partners, the evaluation reveals several concrete advantages for businesses seeking to automate the generation of DPP data from unstructured sensor data. From an economic perspective, these benefits arise primarily from the pipeline's modular, usage-driven design, which allows update frequencies, data granularity, and deployment configurations to be aligned with operational and regulatory requirements.

1. **Automated data harmonization:** The LLM-based mapping reliably converts heterogeneous machine data into standardized formats (> 99% on simple and moderate difficulties & > 93% on complex examples), reducing manual integration effort and ensuring consistent data quality. The schema is strictly enforced through `pydantic` validation, and semantic consistency can be achieved through plausibility checks embedded into SCs.
2. **Regulatory compliance and traceability:** The blockchain layer provides immutable, verifiable records aligned with forthcoming regulatory requirements, simplifying audits and sustainability reporting.
3. **Modular system design:** The architecture's modular structure allows firms to interchange components such as the LLM provider, data storage, cloud or on-premise deployment, and prompt configuration according to their operational, privacy, or regulatory requirements. This flexibility supports long-term maintainability and reduces vendor dependency risks.
4. **Flexible transparency:** The modular blockchain design enables adjustable data granularity, allowing firms to meet regulatory obligations while supporting finer traceability when strategically valuable.

6.4. Limitations

Despite proposing a working end-to-end system architecture, several limitations of this study should be acknowledged, many of which point to promising directions for future research.

First, the official DPP schema has not yet been released. Although the proposed dataset and implementation align with anticipated regulatory requirements, future deviations in the finalized schema may

require adjustments and revalidation. Consequently, the current implementation focuses on mapping and storing selected fragments of environmental manufacturing data rather than constructing a complete DPP across all relevant dimensions, a limitation that can be addressed by extending the pipeline once official specifications become available.

Second, the evaluation relies on a carefully designed synthetic dataset rather than data from a live industrial production environment. Although the dataset is designed to reflect realistic heterogeneity, temporal structure, and schema incompleteness, it cannot fully capture real-world effects such as sensor noise, missing values, or undocumented vendor-specific behavior. In operational factory settings, these factors may affect the robustness and latency of the LLM-based mapping pipeline. Extending the evaluation to real production data, potentially through industry partnerships or pilot deployments, represents a natural next step to validate the system under real-world conditions.

Third, the system was only evaluated using the Waves blockchain, and alternative platforms may offer different trade-offs in terms of scalability, security, or integration flexibility. While GPT-4o consistently delivered the highest mapping accuracy, it remains a proprietary model with associated costs and privacy concerns. Local open-weight alternatives, although preferable for on-premise deployment, did not achieve sufficient performance for complex transformation tasks. The evaluation was also limited to three open-weight models, leaving a broader space of alternatives unexplored and open for future investigation.

Fourth, the environmental and computational footprints associated with the use of LLMs and blockchain technology were not evaluated in this study but represent important considerations for future implementations. The impact of the LLM-based mapping component depends strongly on the selected model, deployment environment (e.g., cloud-based or on-premises), and mapping frequency. Likewise, the environmental implications of the blockchain layer depend on the underlying consensus mechanism. When implemented on non-proof-of-work blockchains, the operational overhead is comparable to conventional distributed server infrastructures.

Finally, prompt engineering was kept to a minimum in this work. More elaborate and adaptive prompt designs may further improve mapping accuracy, especially for open-weight models, and offer an additional lever for reducing the performance gap observed in this study.

6.5. Future work

While the presented system already demonstrates robust end-to-end performance for LLM-based data transformation and on-chain persistence, several directions remain for future research and development.

Fine-tuning for domain specificity: One key direction is the fine-tuning of open-weight LLMs on domain-specific data to close the accuracy gap observed between proprietary models like GPT-4o and open alternatives. By training on representative examples of real-world machine records and their corresponding schema-aligned transformations, such models could learn more stable and semantically faithful mappings even in high-complexity scenarios. This would not only reduce reliance on commercial APIs but also offer greater control over model behavior and data privacy.

Advanced prompting strategies: Beyond fine-tuning, further improvements in prompt engineering could significantly enhance mapping performance. The current results suggest that few-shot prompts with embedded schemas yield the best performance. However, more sophisticated techniques, such as multi-step reasoning prompts, task decomposition, or self-verification chains, may deliver higher semantic accuracy. Prompt ensembling or confidence-weighted fallback strategies may also help mitigate individual prompt failures.

Additional input formats: This work focuses on JSON-to-JSON mapping, whereas real-world sensor outputs often involve different formats such as CSV, XML, or binary data. Future work could extend the pipeline to support such inputs and further explore multi-agent architectures, where each machine is paired with a dedicated LLM agent tailored to its specific data format.

Cross-chain deployment: From a systems perspective, it would be valuable to explore deploying the architecture on alternative blockchain platforms beyond Waves. Owing to the modular design of the blockchain layer, the underlying platform can be readily replaced without requiring changes to the overall system architecture. Options such as Ethereum or Solana offer different performance trade-offs in terms of transaction speed, cost, and developer tooling. Comparative evaluations could identify the most suitable environments for large-scale industrial deployments of DPPs and reveal bottlenecks in portability within the current SC logic.

Data encryption and privacy: Future iterations of the system should explore data encryption strategies to safeguard sensitive production data. While current experiments store mappings in plaintext for verifiability, many industrial use cases demand confidentiality guarantees. Homomorphic encryption, attribute-based encryption, or hybrid off-chain/on-chain encryption schemes could be investigated as means to strike a balance between transparency and privacy (Carls et al., 2023). Moreover, the integration of Zero-Knowledge Proofs may enable validation of schema compliance without revealing the actual content of the mapped records, opening the door to privacy-preserving auditability.

Deeper assessment of economic viability: This study primarily focused on technical performance and mapping accuracy, rather than economic aspects. Case studies across different industrial contexts could provide empirical evidence on integration effort, operational expenditure, and scalability, offering a more comprehensive view of the economic implications of adopting such systems.

In summary, the pipeline presented in this work offers a robust basis for integrating LLM-based data mapping with blockchain-backed DPPs. Still, its long-term viability for industrial deployment will depend on (1) future regulatory requirements, (2) advancements in open-model performance, prompt design, privacy integration, and blockchain interoperability, and (3) the heterogeneity of operational data across manufacturing companies.

6.6. Recommendations for practitioners

The evaluation highlights several practical considerations for industrial actors seeking to automate DPP generation through LLM mapping and blockchain integration.

First, model selection should be guided by data complexity. The experiments show that open-weight models could solve tasks with simple and medium complexity, but were insufficient for complex datasets. Organizations handling data of similar heterogeneity and granularity may currently require proprietary models such as GPT-4o to achieve reliable mappings. However, as open-weight models continue to improve in reasoning capability and domain adaptation, they may soon offer a more sustainable and cost-effective alternative.

Second, cost and dependency risks must be carefully assessed when relying on proprietary APIs, as they introduce recurring costs, privacy constraints, and vendor dependencies. Overall cost scales primarily with mapping frequency, which is ultimately determined by data granularity under forthcoming ESPR delegated acts. However, economic viability can still be influenced through appropriate management of processing frequency, for example, through batching, caching, or hybrid deployment strategies.

Third, system design should emphasize modularity and schema adaptability. Given that the official DPP schema is still under development, practitioners should implement flexible, schema-driven data architectures that can accommodate future regulatory updates. Using intermediate data representations can ensure long-term interoperability and prevent costly redesigns once formal standards are published.

Fourth, the blockchain component of the proposed architecture is designed for full modularity, enabling arbitrary data granularity

Table A.6
Description and example of each perturbation function used to generate a source sample from its corresponding ground truth in the simple and moderate datasets. The complex dataset further increases the variability of mapping tasks.

Machine	Description	Example (Target → Source)
M_1	Different field names with correct units indicated in the field names.	`operation_hours: 10 → UptimeHours: 10`
M_2	Different field names with different units expressed in the field names.	`operation_hours: 10 → UptimeSeconds: 36,000`
M_3	Different field names with correct units embedded in the values.	`operation_hours: 10 → Uptime: ''10 h''`
M_4	Different field names with different units embedded in the values.	`operation_hours: 10 → Uptime: ''36,000 s''`
M_5	Original field names, but different units expressed in the field names.	`operation_hours: 10 → operation_seconds: 36,000`
M_6	Original field names, but different units embedded in the values.	`operation_hours: 10 → operation: ''36,000 s''`
M_7	Original field names with correct units embedded in the values.	`operation_hours: 10 → operation: ''10 h''`
M_8	Identity mapping with original field names and correct units in field names.	`operation_hours: 10 → operation_hours: 10`
M_9	Retains original field names with values and different units nested in sub-objects.	`operation_hours: 10 → operation: {value: 36,000, unit: ''s''}`
M_{10}	Retains original field names with values and correct units nested in sub-objects.	`operation_hours: 10 → operation: {value: 10, unit: ''h''}`

through one or multiple aggregation SCs. Even if forthcoming regulations mandate only coarse-grained reporting, firms may leverage fine-grained aggregation to enable precise and verifiable sustainability assessments, certification processes, and internal performance analytics.

6.7. Recommendations for policy makers

While upcoming DPP-related standards are expected to define data exchange, storage, lifecycle management, interoperability, and integrity requirements, their technical specificity remains uncertain. The actual DPP data schemas will be established later through delegated acts for individual product groups, creating a transitional gap for implementers and complicating the design of interoperable, long-lived data infrastructures.

A particularly critical aspect is the definition of data granularity. Whether information must be recorded at the product, batch, or process level will directly influence system architecture, data volume, and operational cost. Therefore, early and unambiguous guidance on required granularity is essential to enable practitioners to develop compliant, scalable solutions and avoid costly redesigns once product-specific schemas are finalized.

Abbreviations

ESPR	Ecodesign for Sustainable Products Regulation
DPP	Digital Product Passport
LLM	Large Language Model
SC	Smart Contract
DID	Decentralized Identifier

CRediT authorship contribution statement

David Rohrschneider: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Marcel Pehlke:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Uwe Handmann:** Supervision, Project administration, Funding acquisition. **Marc Jansen:** Supervision, Project administration, Funding acquisition.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used ChatGPT by OpenAI in order to improve language and readability. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Funding

This work has been co-funded by the European Commission and the Ministry of Economy, Innovation, Digitization, and Energy of the State of North Rhine–Westphalia within the projects Circular Performer Emscher-Lippe and easy.dpp.nrw.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Dataset

A.1. Perturbation functions

See Table A.6.

A.2. Full examples


```
{
    "source": {
        "CurrentDate": "2024-01-01",
        "UptimeHours": 8.12,
        "PowerKWh": 490.14,
        "SubstanceUsedKg": 619.6,
        "SubstanceWasteKg": 41.97,
        "CarbonDioxideKg": 231.2,
        "WaterUsageLiters": 862.4,
```

```
        "WaterReclaimedLiters": 423.23,
        "YieldUnits": 373
    },
    "target": {
        "date": "2024-01-01",
        "operation_hours": 8.12,
        "energy_consumption_kWh": 490.14,
        "material_used_kg": 619.6,
        "material_waste_kg": 41.97,
        "CO2_emissions_kg": 231.2,
        "water_consumption_liters": 862.4,
        "water_recycled_liters": 423.23,
        "product_output_units": 373
    }
}
```


Listing 1 Sample from the simple dataset and machine 1.


```
{
    "source": {
        "CurrentDate": "2024-01-01",
        "UptimeHours": 8.12,
        "PowerKWh": 490.14,
        "SubstanceUsedKg": 619.6,
        "SubstanceWasteKg": 41.97,
        "CarbonDioxideKg": 231.2,
        "WaterUsageLiters": 862.4,
        "WaterReclaimedLiters": 423.23,
        "YieldUnits": 373,
        "TempC": 76.01,
        "MoisturePercent": 52.08,
        "VibLvlMmps": 1.04,
        "RegenerativePowerPercent": 59.4,
        "IdleMinutes": 34,
        "NoiseDb": 74,
        "NumLocalEmployees": 2,
        "Oiling": "high",
        "Cooling": "operational",
        "Service": true,
        "Fuel": "renewable_fuel"
    },
    "target": {
        "date": "2024-01-01",
        "operation_hours": 8.12,
        "energy_consumption_kWh": 490.14,
        "material_used_kg": 619.6,
        "material_waste_kg": 41.97,
        "CO2_emissions_kg": 231.2,
        "water_consumption_liters": 862.4,
        "water_recycled_liters": 423.23,
        "product_output_units": 373,
        "operating_temperature_C": 76.01,
        "ambient_humidity_percent": 52.08,
        "vibration_level_mmps": 1.04,
        "renewable_energy_percent": 59.4,
        "downtime_minutes": 34,
        "noise_level_dB": 74,
        "worker_count": 2,
        "lubrication_level": "high",
        "cooling_system_status": "operational",
        "maintenance_required": true,
        "fuel_type": "renewable_fuel"
    }
}
```


Listing 2 Sample from the moderate dataset and machine 1.


```
{
    "source": {
        "data_csv": "RegenerativePowerFraction,
            WaterReclaimedMilliliters, TempK,
            SubstanceUsedGrams,
            YieldUnitsThousands, Service,
            Cooling, Oiling, CarbonDioxideTonnes
            , NumLocalEmployees, UptimeSeconds,
            CurrentDateIso, Fuel,
            SubstanceWasteMilligrams,
            VibLvlMmpmin, WaterUsageMilliliters,
             IdleSeconds, NoiseDb,
            MoistureFraction, PowerMWh\n
            \"0.4065\", \"510390.0\",
            \"348.72\", \"561810.0\", \"0.236\",
             \"no\", \"ok\", \"reduced\",
            \"0.27198\", \"3\", \"30600.0\",
            \"2024-01-01T00:00:00\", \"
            dual_source\", \"37600000.0\",
            \"78.6\", \"985580.0\", \"840\",
            \"90\", \"0.5343\", \"0.37674\""
    },
    "target": {
        "date": "2024-01-01",
        "operation_hours": 8.5,
        "energy_consumption_kWh": 376.74,
        "material_used_kg": 561.81,
        "material_waste_kg": 37.6,
        "CO2_emissions_kg": 271.98,
        "water_consumption_liters": 985.58,
        "water_recycled_liters": 510.39,
        "product_output_units": 236,
        "operating_temperature_C": 75.57,
        "ambient_humidity_percent": 53.43,
        "vibration_level_mmps": 1.31,
        "renewable_energy_percent": 40.65,
        "downtime_minutes": 14,
        "noise_level_dB": 90,
        "worker_count": 3,
        "lubrication_level": "low",
        "cooling_system_status": "operational",
        "maintenance_required": false,
        "fuel_type": "hybrid"
    }
}
```


Listing 3 Sample from the complex dataset and machine 8.

Appendix B. Prompt templates

```
System: You are a data transformation assistant
    that processes raw telemetry data collected
    daily from industrial production machines.
Your job is to convert each raw input JSON into
    a structured, standardized JSON object with
    the following goals:

- Extract and rename fields according to the
    target format.
- Ensure units are correct and standardized (e.g
    ., converting g to kg when necessary).
- Ensure the data types are correct (e.g.,
    converting strings to numbers).
- Return only valid JSON - no extra commentary.
```

Table C.7
Pairwise Mann–Whitney U Test results for model comparisons across difficulty levels.

Difficulty level	Model A	Model B	p-Value
Complex	Codestral-22B	DeepSeek-R1-14B	0.545
Complex	Codestral-22B	Gemma-3-12B	0.473
Complex	Codestral-22B	GPT-4o	**0.017**
Complex	DeepSeek-R1-14B	Gemma-3-12B	0.186
Complex	DeepSeek-R1-14B	GPT-4o	**0.005**
Complex	Gemma-3-12B	GPT-4o	**0.001**
Moderate	Codestral-22B	DeepSeek-R1-14B	0.850
Moderate	Codestral-22B	Gemma-3-12B	0.427
Moderate	Codestral-22B	GPT-4o	**0.041**
Moderate	DeepSeek-R1-14B	Gemma-3-12B	0.850
Moderate	DeepSeek-R1-14B	GPT-4o	**0.001**
Moderate	Gemma-3-12B	GPT-4o	**0.041**
Simple	Codestral-22B	DeepSeek-R1-14B	0.273
Simple	Codestral-22B	Gemma-3-12B	0.473
Simple	Codestral-22B	GPT-4o	**0.045**
Simple	DeepSeek-R1-14B	Gemma-3-12B	0.273
Simple	DeepSeek-R1-14B	GPT-4o	0.096
Simple	Gemma-3-12B	GPT-4o	**0.038**

Table C.8
Shapiro–Wilk Normality Test Results for Each Model and Difficulty Tier. All tests are based on the runs with the best configuration per difficulty, as shown in Table 1. Values in **bold** indicate significant deviation from normality ($p < 0.05$).

Difficulty level	Model	W Statistic	p-Value
Simple	Codestral-22B	0.687	**0.0011**
Simple	DeepSeek-R1-14B	0.821	**0.026**
Simple	Gemma-3-12B	0.719	**0.0011**
Simple	GPT-4o	0.366	**1.7e−05**
Moderate	Codestral-22B	0.712	**0.0012**
Moderate	DeepSeek-R1-14B	0.821	**0.026**
Moderate	Gemma-3-12B	0.659	**4.6e−04**
Moderate	GPT-4o	0.366	**1.7e−05**
Complex	Codestral-22B	0.789	**0.0107**
Complex	DeepSeek-R1-14B	0.884	0.146
Complex	Gemma-3-12B	0.737	**0.0025**
Complex	GPT-4o	0.557	**1.7e−05**

```
The output should be formatted as a JSON
    instance that conforms to the JSON schema
    below.

As an example, for the schema {"properties": {"
    foo": {"title": "Foo", "description": "a
    list of strings", "type": "array", "items":
    {"type": "string"}}}, "required": ["foo"]}
    the object {"foo": ["bar", "baz"]} is a well
    -formatted instance of the schema. The
    object {"properties": {"foo": ["bar", "baz
    "]}} is not well-formatted.

Here is the output schema:
```
<<Target Schema>>
```
Human: {input_json}
```

Listing 4 Zero-shot prompt template

```
- Extract and rename fields according to the
    target format.
- Ensure units are correct and standardized (e.g
    ., converting g to kg when necessary).
- Ensure the data types are correct (e.g.,
    converting strings to numbers).
- Return only valid JSON - no extra commentary.

Here are some examples:

Human: <<Source JSON 1>>
AI: <<Target JSON 1>>
Human: <<Source JSON 2>>
AI: <<Target JSON 2>>
Human: <<Source JSON 3>>
AI: <<Target JSON 3>>
Human: <<Source JSON 4>>
AI: <<Target JSON 4>>
Human: {input_json}
```

Listing 5 Few-shot prompt template

```
System: You are a data transformation assistant
    that processes raw telemetry data collected
    daily from industrial production machines.
Your job is to convert each raw input JSON into
    a structured, standardized JSON object with
    the following goals:
```

```
System: You are a data transformation assistant
    that processes raw telemetry data collected
    daily from industrial production machines.
Your job is to generate a python function that
    maps a raw input JSON into a structured,
    standardized JSON object.
```

```
The function should:
- be named ‘map_raw_to_standard‘.
- take a single argument, ‘input_json‘, which is
    a dictionary.
- return a dictionary that represents the
    transformed data.
- extract and rename fields according to the
    target format.
- ensure units are correct and standardized (e.g
    ., converting g to kg when necessary).
- ensure the data types are correct (e.g.,
    converting strings to numbers).

The function output dict should conform to the
    JSON schema below.

As an example, for the schema {"properties": {"
    foo": {"title": "Foo", "description": "a
    list of strings", "type": "array", "items":
    {"type": "string"}}}, "required": ["foo"]}
the object {"foo": ["bar", "baz"]} is a well-
    formatted instance of the schema. The object
     {"properties": {"foo": ["bar", "baz"]}} is
    not well-formatted.

Here is the output schema:
‘‘‘
<<Target Schema>>
‘‘‘

Provide only the function definition inside a
    markdown python code block and without any
    additional comments or explanations.
Human: {input_json}
```

Listing 6 Mapping-function prompt template

Appendix C. Evaluation

See Tables C.7 and C.8.

Data availability

Code and dataset are publicly available. URLs are provided in the article.

References

Abdel-Aty, T. A., Doliman, B., Negri, E., Brooks, S., McFarlane, D., & Macchi, M. (2025). Evaluation of the digital product passport for remanufacturing: A case study using asset administration shell. *IFAC-PapersOnLine*, *59*(10), 867–872. http://dx.doi.org/10.1016/j.ifacol.2025.09.147.

Abreu, H., Pereira, V., & Barata, J. (2025). Blockchain-based digital product passport: Design principles and demonstration. *International Journal of Production Research*, *63*(16), 5863–5882. http://dx.doi.org/10.1080/00207543.2025.2464161.

Adisorn, T., Tholen, L., & Götz, T. (2021). Towards a digital product passport fit for contributing to a circular economy. *Energies*, *14*(8), 2289. http://dx.doi.org/10.3390/en14082289.

Berenguer, A., Morejón, A., Tomás, D., & Mazón, J. N. (2024a). Leveraging large language models for sensor data retrieval. *Applied Sciences*, *14*(6), 2506. http://dx.doi.org/10.3390/app14062506.

Berenguer, A., Morejón, A., Tomás, D., & Mazón, J. N. (2024b). Using large language models to enhance the reusability of sensor data. *Sensors*, *24*(2), 347. http://dx.doi.org/10.3390/s24020347.

Carls, V., Schmidt, L., & Jansen, M. (2023). Overview of multiple user encryption for exchange of private data via blockchains. In J. Prieto, F. L. Benítez Martínez, S. Ferretti, D. Arroyo Guardeño, & P. Tomás Nevado-Batalla (Eds.), *Blockchain and applications, 4th international congress* (pp. 447–453). Cham: Springer International Publishing.

Chase, H. (2022). Langchain (version 0.3.65) [Computer software]. URL: https://github.com/langchain-ai/langchain, MIT License.

Christidis, K., & Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. *IEEE Access*, *4*, 2292–2303. http://dx.doi.org/10.1109/ACCESS.2016.2566339.

Colvin, S., Jolibois, E., Ramezani, H., Garcia Badaracco, A., Dorsey, T., Montague, D., Matveenko, S., Trylesinski, M., Runkle, S., Hewitt, D., Hall, A., & Plot, V. (2025). Pydantic (version 2.11.7) [Computer software]. URL: https://github.com/pydantic/pydantic, MIT License.

Dasaklis, T. K., Voutsinas, T. G., Tsoulfas, G. T., & Casino, F. (2022). A systematic literature review of blockchain-enabled supply chain traceability implementations. *Sustainability*, *14*(4), 2439. http://dx.doi.org/10.3390/su14042439.

DeepSeek-AI (2025). DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning (preprint). http://dx.doi.org/10.48550/ARXIV.2501.12948, arXiv.

Esmaeilian, B., Sarkis, J., Lewis, K., & Behdad, S. (2020). Blockchain for the future of sustainable supply chain management in Industry 4.0. *Resources, Conservation And Recycling*, *163*, Article 105064. http://dx.doi.org/10.1016/j.resconrec.2020.105064.

European Commission (2023). Consolidated text: Regulation (EU) 2023/1542 of the European parliament and of the council of 12 july 2023 concerning batteries and waste batteries, amending directive 2008/98/EC and regulation (EU) 2019/1020 and repealing directive 2006/66/EC. *OJ, L 191*.

European Commission (2024a). Regulation (EU) 2024/1781 of the European parliament and of the council of 13 june 2024 establishing a framework for the setting of ecodesign requirements for sustainable products, amending directive (EU) 2020/1828 and regulation (EU) 2023/1542 and repealing directive 2009/125/EC. *OJ, L 1781*.

European Commission (2024b). Regulation (EU) 2024/3110 of the European parliament and of the council of 27 november 2024 laying down harmonised rules for the marketing of construction products and repealing regulation (EU) no 305/2011. *OJ, L 3110*.

Gemma Team (2025). Gemma 3 technical report. http://dx.doi.org/10.48550/ARXIV.2503.19786, arXiv.

Gleim, L., Pennekamp, J., Liebenberg, M., Buchsbaum, M., Niemietz, P., Knape, S., Epple, A., Storms, S., Trauth, D., Bergs, T., Brecher, C., Decker, S., Lakemeyer, G., & Wehrle, K. (2020). FactDAG: Formalizing data interoperability in an internet of production. *IEEE Internet Things Journal*, *7*(4), 3243–3253. http://dx.doi.org/10.1109/JIOT.2020.2966402.

Götz, T., Berg, H., Jansen, M., Adisorn, T., Cembrero, D., Markkanen, S., & Chowdhury, T. (2022). *Digital product passport : the ticket to achieving a climate neutral and circular European economy?*: *Technical report*, (p. 35). Cambridge: University of Cambridge Institute for Sustainability Leadership, URL: http://nbn-resolving.de/urn:nbn:de:bsz:wup4-opus-80497.

Greiner, M., Seidenfad, K., Langewisch, C., Hofmann, A., & Lechner, U. (2024). The digital product passport: Enabling interoperable information flows through blockchain consortia for sustainability. In F. Phillipson, G. Eichler, C. Erfurth, & G. Fahrnberger (Eds.), *Innovations for community services*: *vol. 2109*, (pp. 377–396). Cham: Springer Nature Switzerland, http://dx.doi.org/10.1007/978-3-031-60433-1_21.

Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Ullah Khan, S. (2015). The rise of “big data” on cloud computing: Review and open research issues. *Information Systems*, *47*, 98–115. http://dx.doi.org/10.1016/j.is.2014.07.006.

Heeß, P., Rockstuhl, J., Körner, M.-F., & Strüker, J. (2024). Enhancing trust in global supply chains: Conceptualizing digital product passports for a low-carbon hydrogen market. *Electronic Markets*, *34*(1), 10. http://dx.doi.org/10.1007/s12525-024-00690-7.

Hojda, M. (2025). Using large languge models for processing sensor data. *Sensors*, *25*(14), 4380. http://dx.doi.org/10.3390/s25144380.

Hulea, M., Miron, R., & Muresan, V. (2024). Digital product passport implementation based on multi-blockchain approach with decentralized identifier provider. *Applied Sciences*, *14*(11), 4874. http://dx.doi.org/10.3390/app14114874.

Jensen, S. F., Kristensen, J. H., Adamsen, S., Christensen, A., & Waehrens, B. V. (2023). Digital product passports for a circular economy: Data needs for product life cycle decision-making. *Sustainable Production and Consumption*, *37*, 242–255. http://dx.doi.org/10.1016/j.spc.2023.02.021.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., & Sayed, W. E. (2023). Mistral 7B. http://dx.doi.org/10.48550/ARXIV.2310.06825, arXiv.

Jung, J., Jeong, E., Song, B., & Watson, K. (2024). Generate digital product passports by integrating production data into the asset administration shell. In *2024 IEEE 29th international conference on emerging technologies and factory automation* (pp. 1–4). Padova, Italy: IEEE, http://dx.doi.org/10.1109/ETFA61755.2024.10711027.

Kayali, M., Lykov, A., Fountalis, I., Vasiloglou, N., Olteanu, D., & Suciu, D. (2024). Chorus: Foundation Models for unified data discovery and exploration. *Proceeding of the VLDB Endowment*, *17*(8), 2104–2114. http://dx.doi.org/10.14778/3659437.3659461.

Khan, M., Wu, X., Xu, X., & Dou, W. (2017). Big data challenges and opportunities in the hype of industry 4.0. In *2017 IEEE international conference on communications* (pp. 1–6). Paris, France: IEEE, http://dx.doi.org/10.1109/ICC.2017.7996801.

Kim, M. J., Han, C. H., Park, K. J., Moon, J. S., & Um, J. (2025). A blockchain-based digital product passport system providing a federated learning environment for collaboration between recycling centers and manufacturers to enable recycling automation. *Sustainability*, *17*(6), 2679. http://dx.doi.org/10.3390/su17062679.

Kühn, M., Baumann, M., Volz, F., & Stojanovic, L. (2025). Digital product passport design supporting the circular economy based on the asset administration shell. *Sustainability*, *17*(3), 969. http://dx.doi.org/10.3390/su17030969.

Langley, D. J., Rosca, E., Angelopoulos, M., Kamminga, O., & Hooijer, C. (2023). Orchestrating a smart circular economy: Guiding principles for digital product passports. *Journal of Business Research*, *169*, Article 114259. http://dx.doi.org/10.1016/j.jbusres.2023.114259.

Lee, M. J. L., Lin, J., & Hsu, L.-T. (2024). Exploring the feasibility of automated data standardization using large language models for seamless positioning. In *2024 14th international conference on indoor positioning and indoor navigation* (pp. 1–6). Kowloon, Hong Kong: IEEE, http://dx.doi.org/10.1109/IPIN62893.2024.10786123.

Leising, E., Quist, J., & Bocken, N. (2018). Circular economy in the building sector: Three cases and a collaboration tool. *Journal of Cleaner Production*, *176*, 976–989. http://dx.doi.org/10.1016/j.jclepro.2017.12.010.

Noaman, M., Khan, M. S., Abrar, M. F., Ali, S., Alvi, A., & Saleem, M. A. (2022). Challenges in integration of heterogeneous internet of things. In X. Zhang (Ed.), *Scientific Programming*, *2022*, 1–14. http://dx.doi.org/10.1155/2022/8626882.

OpenAI (2023). GPT-4 technical report (preprint). http://dx.doi.org/10.48550/ARXIV.2303.08774, arXiv.

Porshnev, S., Ponomareva, O., Borodin, A., & Mirvoda, S. (2019). Problems and methods for integrating heterogeneous data (on example, metallurgical production). In *2019 international multi-conference on industrial engineering and modern technologies* (pp. 1–5). Vladivostok, Russia: IEEE, http://dx.doi.org/10.1109/FarEastCon.2019.8934336.

Pourjafarian, M., Plociennik, C., Rimaz, M. H., Stein, P., Vogelgesang, M., Li, C., Knetsch, S., Bergweiler, S., & Ruskowski, M. (2023). A multi-stakeholder digital product passport based on the asset administration shell. In *2023 IEEE 28th international conference on emerging technologies and factory automation* (pp. 1–8). Sinaia, Romania: IEEE, http://dx.doi.org/10.1109/ETFA54631.2023.10275715.

Queiroz, M. M., Telles, R., & Bonilla, S. H. (2019). Blockchain and supply chain management integration: A systematic review of the literature. *Supply Chain Management: An International Journal*, *25*(2), 241–254. http://dx.doi.org/10.1108/SCM-03-2018-0143.

Saberi, S., Kouhizadeh, M., Sarkis, J., & Shen, L. (2019). Blockchain technology and its relationships to sustainable supply chain management. *International Journal of Production Research*, *57*(7), 2117–2135. http://dx.doi.org/10.1080/00207543.2018.1533261.

Santos, A., Pena, E. H. M., Lopez, R., & Freire, J. (2025). Interactive data harmonization with LLM agents (preprint). http://dx.doi.org/10.48550/ARXIV.2502.07132, arXiv.

Sayogo, D. S., Zhang, J., Luna-Reyes, L., Jarman, H., Tayi, G., Andersen, D. L., Pardo, T. A., & Andersen, D. F. (2015). Challenges and requirements for developing data architecture supporting integration of sustainable supply chains. *Information Technology Management*, *16*(1), 5–18. http://dx.doi.org/10.1007/s10799-014-0203-3.

Shi, D., Meyer, O., Oberle, M., & Bauernhansl, T. (2025). Dual data mapping with fine-tuned large language models and asset administration shells toward interoperable knowledge representation. *Robotics and Computer-Integrated Manufacturing*, *91*, Article 102837. http://dx.doi.org/10.1016/j.rcim.2024.102837.

Simchi-Levi, D., Mellou, K., Menache, I., & Pathuri, J. (2026). Large language models for supply chain decisions. In M. C. Cohen, & T. Dai (Eds.), *AI in supply chains*: *vol. 27*, (pp. 93–104). Cham: Springer Nature Switzerland, http://dx.doi.org/10.1007/978-3-032-07054-8_7.

Srivastava, S. K., Routray, S., Bag, S., Gupta, S., & Zhang, J. Z. (2024). Exploring the potential of large language models in supply chain management: A study using big data. *Journal of Global Information Management*, *32*(1), 1–29. http://dx.doi.org/10.4018/JGIM.335125.

Sudusinghe, J. I., & Seuring, S. (2022). Supply chain collaboration and sustainability performance in circular economy: A systematic literature review. *International Journal of Production Economics*, *245*, Article 108402. http://dx.doi.org/10.1016/j.ijpe.2021.108402.

Voulgaridis, K., Lagkas, T., Angelopoulos, C. M., Boulogeorgos, A.-A. A., Argyriou, V., & Sarigiannidis, P. (2024). Digital product passports as enablers of digital circular economy: A framework based on technological perspective. *Telecommunication Systems*, *85*(4), 699–715. http://dx.doi.org/10.1007/s11235-024-01104-x.

Xia, Y., Xiao, Z., Jazdi, N., & Weyrich, M. (2024). Generation of asset administration shell with large language model agents: Toward semantic interoperability in digital twins in the context of industry 4.0. *IEEE Access*, *12*, 84863–84877. http://dx.doi.org/10.1109/ACCESS.2024.3415470.

Zeid, A., Sundaram, S., Moghaddam, M., Kamarthi, S., & Marion, T. (2019). Interoperability in smart manufacturing: Research challenges. *Machines*, *7*(2), 21. http://dx.doi.org/10.3390/machines7020021.

Zhang, A., & Seuring, S. (2024). Digital product passport for sustainable and circular supply chain management: A structured review of use cases. *International Journal of Logistics Research and Applications*, *27*(12), 2513–2540. http://dx.doi.org/10.1080/13675567.2024.2374256.