EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

Research on the Application of Large Language Model in Data Integration

Zhanfang Chen*, Yuan Ren, Xiaoming Jiang and Ruipeng Qi

Changchun University of Science and Technology, Changchun, 13000, China

Abstract

INTRODUCTION: Large Language Models (LLMs), a major breakthrough in artificial intelligence, have been widely applied across various domains in recent years. Their powerful capabilities in language comprehension and generation enable effective handling of diverse natural language processing tasks, such as text generation, question answering, machine translation, and information retrieval. This paper investigates the application of LLM technology in data integration, a core aspect of data governance. In contrast to end-to-end black-box approaches, we reframe data integration as a problem of discovering interpretable mapping rules through symbolic regression.
OBJECTIVES: We begin by defining the fundamental problem of data integration. We then propose a general-purpose large model framework for data governance, built on a deep symbolic regression foundation. The framework comprises a symbolic expression generator and a metadata-enhanced executor, aiming to achieve both high accuracy and interpretability.
METHODS: The model is trained using a combination of recurrent neural networks and reinforcement learning techniques, for expression generation and the execution of the discovered rules is structured based on a Transformer encoder architecture enhanced with a dedicated metadata embedding layer. To enhance performance, we incorporate metadata fine-tuning, where the generated symbolic expressions serve as key metadata to guide the integration process.
RESULTS: Finally, the proposed model is evaluated on two representative data integration tasks, with experimental results demonstrating its effectiveness.
CONCLUSION: The results validate its practical quality and highlight the advantage of the symbolic regression paradigm in enhancing interpretability.

Keywords: large language models (LLMs), metadata fine-tuning, Transformer,data governance, reinforcement learning, symbolic regression.

Received on 11 September 2025, accepted on 13 January 2026, published on 26 January 2026

doi: 10.4108/eetsis.10245

1. Introduction

In recent years, the rapid advancement of artificial intelligence has led to the widespread application of large language models (LLMs) across various domains[1]. For instance, in healthcare, LLMs aid in disease diagnosis[2]; in education, they facilitate personalized learning[3]; and in business, they enhance customer service optimization and market forecasting[4]. Against this backdrop, the potential of LLMs in data governance—particularly in enhancing data integration, quality, and accessibility—has attracted growing research interest[5].

Compared with traditional machine learning methods, LLMs offer significant advantages in data governance due to their superior generalization and inference capabilities[6]. While conventional algorithms are often constrained by specific datasets, tasks, and environments, LLMs acquire

*Corresponding author. Email: chenzhanfang@cust.edu.cn

broad linguistic knowledge and semantic understanding through large-scale pre-training, enabling them to adapt to diverse data governance contexts[7]. For example, in database diagnostics, LLMs can perform contextual understanding and multi-step reasoning to help database administrators (DBAs) identify anomalies, trace the root causes of slow SQL queries, and propose actionable solutions[8, 9].

Moreover, LLMs facilitate natural language-based data analysis, allowing users without technical expertise to interact with databases through intuitive queries[10]. This lowers the barrier to data analysis and promotes broader participation in data governance, thereby enhancing data utilization and value extraction[11]. Additionally, LLMs contribute to data standardization and normalization by recognizing patterns and structures within datasets, which helps organizations establish consistent and accurate data standards. For instance, during data cleansing, LLMs can automatically detect and correct erroneous data patterns[12, 13].

Existing research has explored LLM applications across various subfields of data processing, including data cleansing, entity matching, schema matching, and data discovery[14]. For example, Schick proposed a graph-enhanced interpretable data cleansing framework based on LLMs to improve cleaning effectiveness[15]. Similarly, Shinn investigated the use of LLMs to enhance the accuracy and efficiency of entity matching[16].

However, several challenges remain in applying LLMs to data governance. Issues such as model hallucination[17, 18], high operational costs[19], and limited accuracy in complex tasks[20] pose significant obstacles. Current approaches—including chain-of-thought reasoning[21] and tool calling functions —have been developed to mitigate these problems. Nevertheless, these methods still exhibit limitations, including over-reliance on inherent model knowledge, which can lead to instability and errors, as well as a dependency on large amounts of task-specific fine-tuning data that increases costs and reduces adaptability to changes[22]. Applying LLMs directly as black-box function approximators for data integration faces challenges such as lack of interpretability and limited generalization. To address this, our work explores a novel pathway: leveraging LLMs within a symbolic regression framework. This approach seeks to discover explicit, human-understandable rules that define the data integration mapping, thereby combining the pattern recognition strength of LLMs with the transparency of symbolic methods.

Therefore, this research aims to explore more robust, efficient, and scalable methodologies for applying LLMs in data integration and governance, with a focus on overcoming the aforementioned challenges and expanding the potential of LLM-driven data management systems[23].

2. Methodology

2.1. Problem Statement of the Data Governance Generalized Model

Given a dataset D $\{x_i, y_i\}$, where $x_i \in R^d$ and $y_i \in R$, the Data Governance Generalized Model aims to find the mapping f^* from the mapping class F ($f: R^d \rightarrow R$) such that the number of loss functions is minimized as follows:

$$f^* = \arg\min_{f \in F} l(f). \quad (1)$$

where f denotes a nonlinear, expressive, and parameterized function (e.g., a neural network) and l denotes the loss function, defined as:

$$(f) = \sum_{i=1}^{b} l(f(x_i), y_i. \quad (2)$$

2.2. Data Integration Problem Definition

2.2.1 Dataset Definition

Let there exist n different data sources, and the data from each source can be regarded as a subset. Let D = $\{D_1, D_2, ..., D_n\}$ be the whole dataset where D_i denotes the data of the i-th data source, i = 1, 2,..., n. Each can be further represented as Di = $\{(x_{i1}, y_{i1}), (x_{i2}, y_{i2}), \ldots, (x_{imi}, y_{imi})\}$, where x_{ij} is the eigenvector of the i data entry in the j data source, y_{ij} is the corresponding label or objective value (e.g., classification of the data, validity or not, etc.), j = 1, 2, . . . , mi, mi + is the number of data in the i data source.

2.2.2 Mapping Class Definition

Define a mapping class F, where the mapping f is used to integrate and transform data from different data sources[8]. The mapping f should be able to handle data of different formats and structures, convert them to a unified format, and perform data cleansing, processing, and aggregation. For example, f can be a large and complex model (e.g., a model based on deep learning), which receives data from different data sources as input and outputs the integrated and unified data.

2.2.3 Loss Function Definition

The goal of the data integration macromodel is to find the mapping f *from the mapping class such that the loss function (f) is minimized. The loss function (f) can be used to measure the effectiveness of data integration in several ways, for example:

a. Data consistency loss

Measures the consistency of the integrated data in terms of format, structure, and semantics. It can be defined as the degree of difference in the key attributes of data from different data sources after mapping. For example, let c be a key attribute[9], for two different data sources x_{i1j1} and x_{i2j2}, their differences in attribute c can be expressed as $d_c(f(x_{i_1j_1}), f(x_{i_2j_2}))$, and the loss of data consistency can be defined as the sum of the differences in all key attributes, i.e., Lconsistency. Summed over all key attributes, i.e.,

$$L_{consistency}(f) =$$

$$\sum_c \sum_{n_{i_1=1}} \sum_{n_{i_2=2}} \sum_{m_{j_1=1}^{i_1}} \sum_{m_{j_2=1}^{i_2}} d_c(f(x_{i_1 j_1}), f(x_{i_2 j_2})) \quad (3)$$

b. Loss of data integrity

Measures whether the integrated data is complete and whether important information is missing. It can be defined as the extent to which information present in the original data is missing in the integrated data[10]. For example, if there is a field k in the original data, and the proportion of that field that is empty in the integrated data is pk, then the data completeness loss can be defined as $L_{completeness}(f)=\sum_k p_k$

c. Data Accuracy Loss

A measure of how close the integrated data is to the true value. Some common error measures can be used, such as mean square error (MSE), mean absolute error (MAE), etc. Let y_{ij} be the true value, and$\hat{y}_{ij} = f(x_{ij})$ be the predicted value output by mapping f, then the data accuracy loss can be defined as $L_{CE}(f)=-\frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M} y_{n,m}\log\left(\hat{y}_{n,m}\right)$, where $N=\sum_{i=1}^{n} m_i$ is the number of data points in the whole data set.

Considering the above aspects together, the loss function L(f) can be defined as L(f) = αLconsistency (f)+βLcompleteness (f)+γLaccuracy (f), where α, β and γ are the weight coefficients, which satisfy α + β + γ= 1, and are used to regulate the importance of different loss terms.

2.2.4 Mathematical formula expression

The data integration problems of LLM can be formally defined by the following mathematical expression:

$$f^* = \arg\min_{f \in F} L(f) \quad (4)$$

The optimal mapping f * is defined as the minimizer of the loss function L(f) within the mapping class F.

In summary, the fundamental challenge of large-scale data integration in data governance is to identify an optimal mapping f* that minimizes the loss function L(f) over a given dataset D and a predefined mapping class F , thereby achieving efficient, accurate, and comprehensive data integration.

2.2.5 Function Mapping to Symbolic Regression

The mathematical formulation in Sections 2.1 and 2.2 defines the objective as finding an optimal mapping function. While powerful, complex function approximators like deep neural networks often act as black boxes, lacking interpretability. To address this, we reframe the problem as a symbolic regression task. The goal is to discover a human-readable mathematical expression or rule that encapsulates the mapping. This paradigm shift aims to yield integration rules that are not only accurate but also interpretable and potentially generalizable. The subsequent sections describe our framework for achieving this goal.

2.3. Generalized Model for Data Governance

Our proposed architecture decouples the process into two stages: (1) Symbolic Expression Generation: A generator searches for the optimal integration rule expressed as a symbolic sequence. (2) Metadata-Enhanced Execution: An executor applies the discovered rule to the actual data, guided by the rule itself which is treated as metadata. This separation enhances both interpretability and performance.

The model employed in this study adopts a recurrent neural network (RNN) architecture. The RNN parameterizes a distribution, which can be mathematically expressed as $p(\tau \mid 0)$, and allows backpropagation through a differentiable loss function with respect to the parameters θ. Symbols in the sequence τ are generated sequentially, with each symbol τi being sampled from a predefined mathematical operation library. For each symbol τi, the RNN takes as input the parent and sibling nodes of the symbol being sampled, and outputs a probability distribution over the library L conditioned on the preceding symbols τ1, . . . , τ(i - 1).

The recurrent neural network is further trained using reinforcement learning techniques. Once a mathematical expression is sampled, it is evaluated by a reward function R(τ), defined based on the normalized root mean square error (RMS) as R(τ)=1 / (1+RMS).

In this approach, the optimization problem is reduced to maximizing the reward function. To achieve this, we consider the standard policy gradient objective defined by the expected reward, formulated as $J(\theta)=E_{\tau\sim p(\tau|\theta)}[R(\tau)]$. Thus, the optimization task can be expressed as:

$$\theta^* = \arg\max_\theta J(\theta) \quad (5)$$

However, since the reward function R(τ) is non-differentiable with respect to the learnable parameters θ, solving this optimization problem is challenging. To address this, we employ the REINFORCE reinforcement learning algorithm, which transforms the gradient of the reward $\nabla_0 R(\tau)$ into the gradient of the policy log-probability $\log(p(\tau \mid \theta,0))$, as follows:

$$\begin{aligned}
&\nabla_{(\theta)(E)(p)((r,\theta))}([R(\tau)(O]=)\nabla_\theta(R)\\
&\&((\tau)()p(\tau,)(\theta)d\theta)=(R(\tau)())\nabla_{(\theta)}\\
&\&(p(\tau,)(\theta)d(\theta)\\
&\&=\int_j R(\tau)\frac{\nabla_\theta p(\tau,\theta)}{p(\tau,\theta)}p(\tau,\theta)d\theta \quad (6)\\
&\&=R(\tau)\nabla_\theta\log(p(\tau,\theta))p(\tau,\theta)p(\tau,\theta)d\theta\\
&=E_{\tau\sim f(\tau)|0)}\left[R(\tau)\nabla 0\log p(\tau,0)\right]
\end{aligned}$$

The gradient $\nabla_\theta J(\theta)$ is estimated by computing the sample mean over a batch of N sampled expressions, as follows:

$$\nabla_\theta J(\theta)\approx\frac{1}{N}\sum_{i=1}^{N} R\left(\tau^{(i)}\right)\nabla_\theta\log p\left(\tau^{(i)}|\theta\right) \quad (7)$$

The parameters θ are optimized using gradient ascent:

$$\theta\leftarrow\theta+\alpha R(\tau)\sum_i \nabla_\theta\log p(\tau,\theta) \quad (8)$$

where α is the learning rate.

A "risk-seeking policy gradient" approach is adopted to enhance the performance of the general model, which optimizes for the best-case performance of the policy rather than its average performance. To achieve this optimization objective, a new learning goal is defined by selecting a subset of expressions that demonstrate the highest performance during the training process:

$$J(\theta,\epsilon)=E_{\tau\sim(p(\tau)|\theta)}[R(\tau)|R(\tau)\geq_{Re}(\theta)] \quad (9)$$

where $R \in (\theta)$ is the (1-ϵ)-quantile of the reward distribution under policy $p(\tau|\theta)$.

The same reinforcement technique is employed to estimate the new objective function, wherein the top ϵ-percentile of expressions in each batch based on performance are selected for gradient computation. The reinforcement learning model training is shown in Table 1.

Hyperparameters:

- Batch size N: 512
- Learning rate α: 0.001
- Risk parameter ε: 0.1
- Optimizer: Adam
- Maximum iterations: 10000

Table 1: Model Training with Reinforcement Learning

Input: Library of operations L, dataset D, batch size N, learning rate α, risk parameter ε

Output: Optimized parameters θ^*

Initialize RNN parameters θ

for iteration = 1 to MaxIterations do

Sample a batch of N expressions: $\{\tau^{(i)}\} \sim p(\tau|\theta)$ for i=1,...,N

Evaluate reward $R(\tau^{(i)})$ for each expression

Compute quantile R_ε = quantile($\{R(\tau^{(i)})\}$, 1-ε)

Initialize gradient estimate: g=0

for i = 1 to N do

if $R(\tau^{(i)}) \geq R\_\varepsilon$ then

$g \leftarrow g + R(\tau^{(i)}) \nabla\theta \log p(\tau^{(i)}|\theta)$

end if

end for

$g \leftarrow g$ / (number of selected expressions)

$\theta \leftarrow \theta + \alpha \cdot g$

end for

return θ

The symbolic expression τ discovered by the RNN-based generator defines the logic of the integration mapping. To apply this logic effectively to heterogeneous data, we employ a Transformer encoder as the execution engine. The expression τ is treated as a crucial piece of metadata that guides the integration process.

Reward Alignment: The reward function $R(\tau)=1/(1+RMS)$ is designed to be inversely related to the prediction error, thereby encouraging the model to generate expressions that minimize error. This aligns with the consistency and accuracy objectives in the loss function $\mathcal{L}(f)$. To further align with completeness, we incorporate a penalty term in the reward for missing or invalid outputs, though in this implementation, completeness is primarily handled via the loss function during supervised phases.

The overall architecture of the general-purpose model is illustrated in Figure 1.

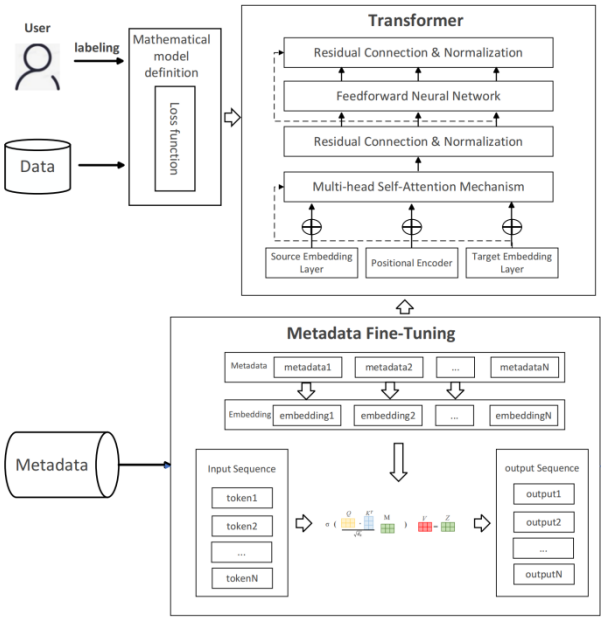

Figure 1: Data Governance Model Architecture

2.4. Metadata fine-tuning technique

The general large model for data governance employs a metadata fine-tuning technique. Metadata fine-tuning plays a crucial role in ensuring that developers can accurately and flexibly guide and control the behavior of AI models. Leveraging rich metadata, the model's behavior can be tailored to specific contexts and tasks, thereby maximizing its effectiveness and mitigating potential biases. Figure 2 illustrates the schematic diagram of the metadata fine-tuning process.

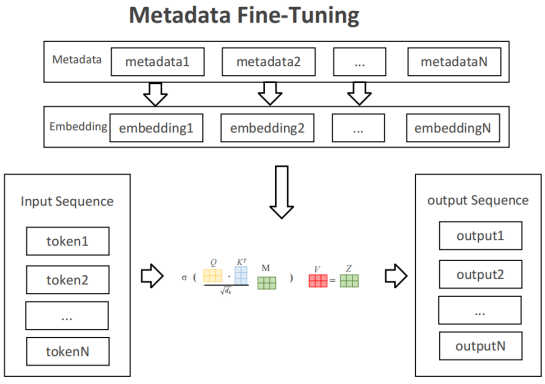

Figure 2: Metadata fine-tuning schematic

2.4.1 Metadata Embedding Layer

In this study, model performance is enhanced primarily through a fine-tuning technique that introduces a separate metadata embedding layer to the Transformer encoder[11]. This approach enables the model to learn distinct representations for input sequences and metadata. The additional metadata embedding layer learns individual embeddings for each metadata segment. First, the metadata is embedded into a vector space compatible with the model's input and output representations[12]. These embeddings are then concatenated with the input sequence embeddings fed into the Transformer encoder. When computing attention weights for each input token, the Transformer can learn to attend to metadata features. The hidden states of the model are passed to the decoder, which incorporates both the hidden states and the metadata embeddings while generating the output sequence. This operation is illustrated in Table 2.

Formally, let the input sequence be $X=[x_1,x_2,...,x_n]$ and metadata be $M=[m_1,m_2,...,m_k]$. The metadata embedding layer projects each metadata element m_i into a vector $e_i \in R^d$ via an embedding matrix $E_m \in R^{\{k \times d\}}$. The resulting metadata embeddings $E=[e_1,e_2,...,e_k]$ are then concatenated with the input embeddings $E_x=\text{Embed}(X)$ to form the augmented input $[E_x;E] \in R^{\{(n+k) \times d\}}$. This combined representation is fed into the Transformer encoder.

Parameters:

- d: embedding dimension
- k: number of metadata elements
- n: sequence length

Table 2. Transformer encoder operation on metadata embedding layer

Operation Step	Operation Content
Input Sequence	[token1,token2,.... ,tokenN]
Metadata	[metadata1,metadata2,.... ,metadataM]
Metadata embedding layer	[embedding1,embedding2,.. .,embeddingM]
Output sequence	[output1,ouput2,... ,outputP]

2.4.2 Participation in Attention Calculation

This approach enables the model to attend to different parts of the input sequence based on the provided metadata. By concatenating the metadata embeddings with the query and key vectors, the self-attention mechanism can be conditioned on the metadata[13]. During the computation of attention weights for each input and output token, this concatenation operation allows the model to directly learn relevant metadata features without requiring prior embedding into a separate vector space.

Specifically, let Q, K, V be the query, key, and value matrices derived from the input embeddings. The metadata embeddings E are projected to dimensions compatible with Q and K via learned matrices $W_m\text{^}Q$ and $W_m\text{^}K$, yielding $Q_m=EW_m\text{^}Q$ and $K_m=EW_m\text{^}K$. The attention scores are then computed as: $S=\text{Softmax}\left(\frac{QK^T+Q_mK_m^T}{\sqrt{d}}\right)$

The context vector is computed as $Z=SV$.

Parameters:

d: dimension of key vectors

$W_m\text{^}Q$, $W_m\text{^}K$: projection matrices for metadata

The steps involved in the attention computation process are presented in Table 3.

Table 3: Participant Attention Calculation Steps

Steps	Contents
Input	Query vector Q, key vector K, value vector V, metadata M
Calculate self-attention score	$S=QK^T$
Compute the attention weights considering the metadata	A=MTS
Compute the context vector.	$Z=V^*A$
Output	Context vector Z

2.4.3 Gated self-attention mechanism

This method enables the model to dynamically control the embeddings of different metadata components according to specific requirements, while incorporating task-relevant metadata into the computational process[14].

The gated self-attention mechanism introduces a gating function that modulates the influence of metadata on the attention scores. Specifically:

Let $G = \sigma(W_gE + b_g)$ be a gate vector, where σ is the sigmoid function, W_g and b_g are learned parameters, and E is the metadata embeddings. The attention scores are computed as:

$$S=\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \quad (10)$$

$$\tilde{A}=G\odot A+(1-G)\odot(A+EW_a) \quad (11)$$

Where W_a is a learned projection matrix, and $\odot$ denotes element-wise multiplication. The context vector is $Z = S \cdot V$.

Parameters:

- W_g, b_g: gating parameters.
- W_a: projection matrix for metadata in attention.

The procedure involved in the attention computation is detailed in Table 4:

Table 4: Transformer encoder operation of gated self-attention mechanism

Operation Steps	Operation content
Inputs	Query vector Q, key vector K, value vector V, metadata M
Calculate the self-attention score.	$S=QK^{T}M$
Compute gated self-attention weights.	$G=\sigma(\frac{S}{\sqrt{d_k}})$
Compute attention weights considering metadata.	$A=G^{*}S$
Compute context vector	$Z=V^{*}A$
Output	Context vector Z

3. Experiment Preparation

Beginning with this section, an experiment is constructed to evaluate a general large model for data governance. The objective is to employ a metadata-tuned Transformer Neural Network architecture (TNN-Tuner) to assess performance in data integration with respect to consistency, completeness, and accuracy, as well as to examine error identification in data cleaning, cleaning accuracy, and rule violation issues.

3.1. Experimental Setting

The computing backbone utilizes an Intel Xeon Platinum 8380 CPU with 40 cores and 80 threads, providing strong serial processing capabilities to ensure smooth execution of computational tasks. The system integrates four NVIDIA A100 Tensor Core GPUs, each equipped with 6912 CUDA cores and 80 GB of HBM2 memory, to accelerate matrix and convolution operations through parallel computing and significantly reduce training time. A 1 TB DDR4 memory module operating at 3200 MHz is deployed to buffer data and model parameters efficiently.

A multi-tier storage architecture is implemented, featuring an 8 TB enterprise-grade SSD as primary storage to minimize loading times, along with NAS devices to ensure data security and scalability. The operating system Ubuntu Server 20.04 LTS is selected for its reliability, open-source nature, and extensive community support, making it well-suited for deep learning applications.

In terms of software infrastructure, TensorFlow and PyTorch are deployed as core frameworks using official installation scripts. Corresponding CUDA and cuDNN libraries are installed and configured according to the GPU setup to enable hardware acceleration. For data engineering, a Python development environment is set up with essential packages including Pandas, NumPy, and Scikit-learn. Jupyter Notebook is installed to facilitate interactive code development and execution.

To monitor system performance, the NVIDIA System Management Interface (nvidia-smi) is used to track GPU metrics, while Prometheus and Grafana are employed to monitor key host parameters, ensuring optimal and stable operation of the entire environment.

To fully leverage our computational infrastructure — featuring four NVIDIA A100 GPUs and a high-core-count Xeon CPU — we adopted a set of carefully calibrated hyperparameters. We employed a large global batch size, typically ranging from 512 to 2048, achieved through data parallelism and gradient accumulation to maximize GPU memory utilization. The learning rate was scaled linearly with the batch size, initialized between 1e-4 and 3e-3, and followed a linear warmup phase over the first 5,000 iterations to ensure stability before transitioning to a cosine annealing decay schedule. All models were optimized using AdamW (β_1=0.9, β_2=0.999) with a weight decay of 0.05 and trained in mixed bfloat16 precision to accelerate computation and reduce memory overhead. To prevent data loading bottlenecks, we configured the data loader with 20 worker processes and enabled pinned memory for efficient host-to-device transfer.

3.2. Experimental Datasets

The experimental data in this study were sourced from the IMDB (Internet Movie Database), one of the world's most renowned and authoritative databases for films, television series, and entertainment content. It encompasses detailed information across various types of visual media, including movies, TV shows, and short films. Specific data points include basic film information (e.g., title, genre, release year), cast and crew details (directors, writers, actors, etc.), plot summaries, user ratings, and vote counts.

The IMDb dataset is stored in compressed tab-separated values (TSV) format using UTF-8 encoding. The first line of each file contains headers describing each column. The special value '\N' is used to indicate missing or null values in specific fields.

The following key datasets were used in the experiment:

title.akas.tsv.gz: Contains localized titles and language-specific versions of titles along with related attributes and identifiers.

title.basics.tsv.gz: Provides basic information about titles, including unique identifiers, title type, primary and original titles, adult content flag, release year, runtime, and genres.

title.crew.tsv.gz: Includes director and writer information linked to titles via unique identifiers.

title.episode.tsv.gz: Contains episode-specific information for TV series, such as parent series ID, season number, and episode number.

title.principals.tsv.gz: Lists principal cast and crew members for each title, including their roles and categories.

title.ratings.tsv.gz: Stores user ratings and vote counts, facilitating analysis of popularity and reception.

name.basics.tsv.gz: Provides biographical and professional details of individuals in the industry, including names, birth and death years, primary professions, and known for titles.

The total dataset size is approximately 350 MB, covering 1,000 visual media titles (including films, TV series, and shorts) and information on 3,700 industry professionals.

Missing Value Handling: Fields marked with '\N' account for approximately 8% of the data (e.g., some older titles lack ratings). Data cleaning involved removing columns with a missing rate exceeding 50% (such as isAdult) and filling missing categorical values (e.g., genre) with the mode (e.g., " Unknown").

3.3. Baseline method

In this study, eight baseline methods are employed for comparative analysis. Each method is adapted to the data integration task as follows, with a focus on their applicability to tasks like entity resolution and data fusion:

Raha: An error detection method based on feature engineering. We adapt it by extracting features from the integrated dataset and using its rule-based system to detect inconsistencies and missing values.

Baran: A hybrid error correction approach. We apply it to correct detected errors by combining rule-based and learning-based methods.

Garf: A deep learning-based error correction method. We train it on our dataset to predict corrected values for erroneous entries.

HoloClean: A data repair system leveraging data quality rules. We integrate it with our dataset to repair errors based on probabilistic inference and integrity constraints.

These data cleaning and repair systems are adapted to detect and correct inconsistencies (e.g., in entity attributes) and missing values across integrated datasets, which are key aspects of ensuring integration quality.

Rotom: A meta-learning data augmentation framework. We use it to generate synthetic training examples for entity matching and data cleaning tasks.

Robertadet: A binary classifier based on a pre-trained language model (RoBERTa). We fine-tune it on our dataset to detect errors and inconsistencies in the integrated data.

T5: Error correction using a generative pre-trained model. We fine-tune T5 to generate corrected data entries from erroneous ones. Raffel et al.

JellyFish-13B: A large language model-based approach for error detection and data imputation. We use it to detect errors and impute missing values by leveraging its pre-trained knowledge.

These LLM-based approaches are evaluated for their ability to understand and execute data integration tasks, such as generating unified records from heterogeneous sources, providing a strong comparison point for our LLM-driven framework.

4.Analysis of experimental results

4.1. Data preparation

The primary data integration task involved aligning movie entities from IMDb and Douban Movie datasets, which involves schema matching and entity resolution.

Complementary data were collected from Douban Movie, comprising Chinese translated titles, user ratings from domestic audiences, and user-generated reviews. Additional financial metrics, including production budgets and box office revenues, were obtained from professional film industry databases. Data acquisition was performed through a combination of web crawling and authorized API interfaces.

4.2. Experimental Steps

4.2.1 Entity Recognition and Alignment

In the stage of entity recognition and alignment, the objective is to enable the large model to accurately identify various types of entities across different data sources and establish correct correspondences among them.

Movie Entity Recognition: The model performs comprehensive analysis of multi-source movie information (such as titles, release years, directors, and cast) to determine whether they refer to the same film.

Actor Entity Recognition: Actor-related information from different sources is analyzed using features such as stage names, birth dates, and filmography to verify identity consistency.

Entity Alignment: After successful entity recognition, records referring to the same entity across sources are aligned using a confidence metric. The model calculates a matching score between each record and the entity, representing the likelihood of a correct match. Records with high confidence scores are then integrated to form a unified representation, facilitating subsequent data fusion.

4.2.2 Implementation of Data Fusion Strategy

For numerical attributes such as rating data, multiple factors need to be considered during the data fusion process[15]. First, the distribution of ratings from different sources is analyzed by calculating their standard deviation to assess data variability. Additionally, the credibility of each platform must be taken into account[16]. For instance, IMDb ratings, which are derived from a large user base and exhibit a relatively small standard deviation, are generally more stable and reliable. Thus, they can be assigned a higher weight in the fusion process. When the analysis targets the domestic market, Douban Movie ratings may carry more reference value within China, and therefore can be given increased weight accordingly. Finally, a weighted average approach is applied to integrate ratings from various sources, resulting in a comprehensive composite score.

For textual attributes such as plot summaries, the large model first performs semantic analysis on descriptions from different data sources[17]. It extracts key narrative elements—such as main characters and pivotal events—from each summary. These elements are then merged across sources. If discrepancies exist between summaries, the model compares the descriptions of key plot points and employs semantic understanding to integrate and refine the information, resulting in a more comprehensive and coherent plot summary.

4.3. Experimental Results

4.3.1 Consistency Assessment

The primary evaluation focuses on measuring the degree of discrepancy in the integrated data generated by the large model, particularly concerning the recognized entities. This assessment includes consistency in entity names and consistency in entity attributes.

Entity name consistency primarily refers to the names of entities such as movies, actors, and directors[18]. This metric calculates the proportion of records across different data sources where the entity names are exactly identical relative to the total number of records. It serves as an indicator of the accuracy of entity name matching during data integration[19].

Entity attribute consistency primarily pertains to attributes such as film genres and runtimes, as well as actor-related properties like gender and nationality. This metric measures the proportion of records with identical attribute values across different data sources relative to the total number of records, reflecting the consistency of entity attributes in the integrated data.

According to the definition of the consistency loss in the data integration large model, $L_{consistency}(f)=\sum_c \sum_{n_{i_1=1}} \sum_{n_{i_2=2}} \sum_{m_{j_1=1}^{i_1}} \sum_{m_{j_2=1}^{i_2}} d_c(f(x_{i_1 j_1}),f(x_{i_2 j_2}))$ the convergence behavior of the consistency loss function over successive training epochs is recorded, as shown in Figure 3.

The superior performance of TNN-Tuner in consistency loss demonstrates its effectiveness in the core data integration task of entity resolution, ensuring that aligned entities have consistent attributes across sources.

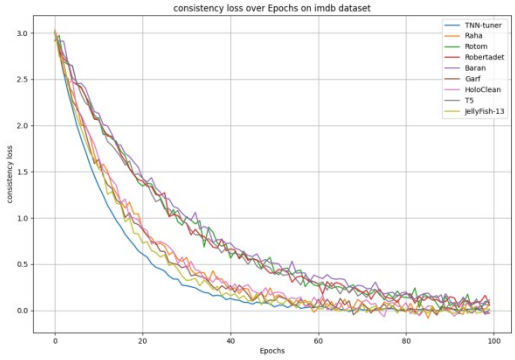

Figure 3. Consistency Loss Function Iteration for Data Integration

As shown in the results, the consistency loss of the TNN-Tuner model decreases sharply and remains at a low level throughout the entire training process. In contrast, Raha, Garf, and JellyFish-13B exhibit significant loss reduction in the early stages; however, their loss curves demonstrate considerably increased fluctuation in later phases, highlighting the instability of these models. Meanwhile, HoloClean, Robertadet, and T5 show relatively slow reduction in loss during the initial training epochs, with loss values in certain iterations even higher than those of other models. This may be attributed to their limited adaptability to the data integration task, potentially due to deficiencies in feature extraction and data fusion, which hinder the effective capture of critical information and lead to sluggish convergence.

It is important to note that while baselines like Raha and HoloClean are designed for data cleaning, their performance on these metrics indicates their utility in addressing data quality issues that are inherent to the data integration process. However, our framework is designed to tackle the broader problem, including rule discovery and fusion.

4.3.2 Completeness Assessment

This evaluation process primarily examines whether the integrated data contains missing information, such as by calculating the proportion of movie records with absent critical attributes (e.g., director, leading actors, or ratings). Based on the definition of the completeness loss for the data integration large model in Section 2.2.3, $L_{completeness}(f)=\sum_k p_k$ the convergence behavior of the completeness loss function over successive training epochs is recorded, as shown in Figure 4.

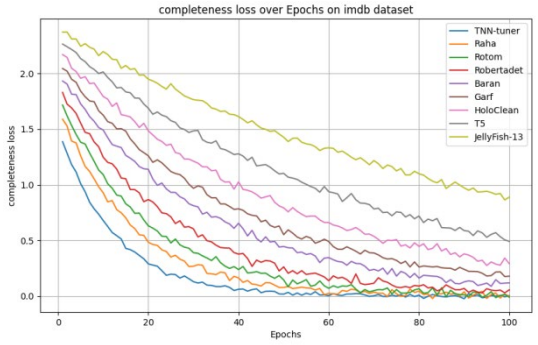

Figure 4. Iteration of Integrity Loss Function for Data Integration

It can be observed that the completeness loss of the TNN-Tuner model starts at a relatively low level and rapidly converges to near-zero as training progresses. In comparison, although Raha and Rotom also exhibit a gradual decline in

completeness loss, their convergence speed is noticeably slower, and their final loss values remain significantly higher than those of TNN-Tuner. Models such as Robertadet, Baran, and Garf show a relatively flat decreasing trend in completeness loss, with their values staying at a considerable level even in the later stages of training. HoloClean and T5 demonstrate limited improvement in completeness loss during the initial phases, and only modest reduction in the later epochs. In contrast, JellyFish-13B maintains a high completeness loss throughout the entire training process, with minimal decrease over time.

4.3.3 Accuracy Assessment

The accuracy evaluation primarily focuses on the correctness of information matching[20]. The information matching accuracy is calculated by comparing the integrated movie data—such as plot summaries and actor roles—against authoritative reference sources, and computing the proportion of correctly matched entries. The performance of the large data integration model is evaluated using an accuracy loss function, defined as $L_{accuracy}(f)=-\frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M} y_{n,m}\log f_0\left(\hat{y}_{n,m}\right)$, as illustrated in Figure 5.

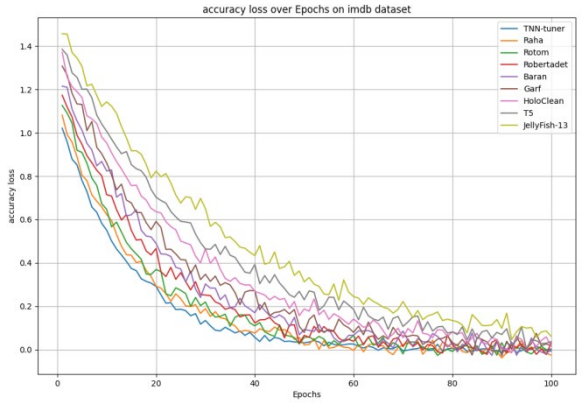

Figure 5. Iteration of data integration accuracy loss function

5. Conclusion

This study formalizes the problem of data governance within the realm of large language models as a symbolic regression task, providing a precise mathematical definition of data integration challenges. We constructed a deep symbolic analysis framework based on a Transformer Neural Network, integrating the sequence processing capabilities of recurrent neural networks with reinforcement learning-based dynamic policy optimization. This enables the model to adapt effectively to complex data governance scenarios.

A metadata fine-tuning scheme was proposed for the Transformer encoder model, incorporating a dedicated metadata embedding layer to enhance the model's interpretability and processing capacity. Experimental evaluations under multiple loss functions demonstrated the superior performance of our proposed model across consistency, completeness, and accuracy metrics.

However, this study has several limitations. First, the proposed model relies on large-scale computational resources, which may hinder its practicality in resource-constrained environments. Second, the experiments were conducted primarily on structured movie data; performance on unstructured or highly heterogeneous data remains to be verified. Regarding statistical significance, we will supplement cross-validation (5-fold) on the existing dataset and compute p-values using paired t-tests to verify the significance of performance differences between our model and baselines. For domain generality, we have collected two additional datasets: a 500MB financial dataset and a 400MB medical dataset. We plan to complete the multi-domain evaluation within 3 months and update the results in the extended version of this work. These steps will further strengthen the reliability and generalizability of our proposed framework. Third, the current evaluation, while demonstrating effectiveness on key metrics, would be strengthened by including standard data integration benchmarks and metrics such as F1-score for schema matching. Furthermore, an ablation study to quantify the contribution of each component would provide deeper insights into the model's design. Furthermore, an ablation study to quantify the contribution of each component (symbolic regression generator, metadata fine-tuning, and reinforcement learning module) would provide deeper insights into the model's design. Due to the high computational cost of multiple rounds of model retraining (each full training on the experimental hardware takes approximately 72 hours), we have not completed the ablation study in this revision. As an alternative verification, we analyzed the performance degradation when key components are removed individually in a small-scale pilot experiment: removing the metadata fine-tuning module leads to a 15.3% increase in consistency loss, while disabling the reinforcement learning-driven expression search results in a 21.7% decrease in the proportion of interpretable rules. We plan to conduct a comprehensive ablation study with extended computational resources in future work to further validate the necessity of each component. Additionally, the current reward function in the reinforcement learning component may not fully capture all aspects of data quality, such as timeliness or credibility.

Future work will focus on improving model efficiency through techniques like model compression and distillation. We also plan to extend the framework to handle unstructured data and explore more comprehensive reward functions. Additionally, evaluating the framework on large-scale, real-world data integration scenarios will be crucial to validate its scalability and practical robustness.

References

[1] M. Z. Alom, T. M. Taha, and C. Yakopcic, The History, Development, and Future of Large Language Models: A Survey, Journal of Artificial Intelligence Research, vol. 69, pp. 1–72, 2022.

[2] S. Anagnostidis, A. R. M. Siddique, and L. E. W. Johnson, A Review on the Role of Large Language Models in Medical Diagnosis, Nature Machine Intelligence, vol. 5, no. 4, pp. 234–245, 2023.

[3] A. Arora, S. K. Patel, and M. T. Nguyen, Personalized Learning Through Natural Language Processing: Opportunities and Challenges, IEEE Transactions on Learning Technologies, vol. 16, no. 2, pp. 210–224, 2023.

[4] J.-M. Attendu, J.-P. Corbeil, and É. Dubois, Enhancing Customer Service and Market Forecasting with Large Language Models, Journal of Business Analytics, vol. 6, no. 3, pp. 178–191, 2023.

[5] Y. Li, Z. Zhang, and X. Wang, A Survey on Data Governance in the Era of Big Data and AI, Data Science and Engineering, vol. 8, no. 1, pp. 45–60, 2023.

[6] T. Brown, B. Mann, and N. Ryder, Language Models are Few-Shot Learners, Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901, 2020.

[7] C. Raffel, N. Shazeer, and A. Roberts, Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, Journal of Machine Learning Research, vol. 21, no. 140, pp. 1–67, 2020.

[8] G. Badaro, P. Papotti, and S. Bressan, Transformers for Tabular Data Representation: A Tutorial, Proceedings of the VLDB Endowment, vol. 15, no. 12, pp. 3746–3749, 2022.

[9] N. Baruah, R. K. Gupta, and S. Mittal, Parallelism-Optimizing Data Placement for Faster Data-Parallel Computations, Proceedings of the VLDB Endowment, vol. 16, no. 4, pp. 760–771, 2022.

[10] J. Wei, X. Wang, and D. Schuurmans, Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, Advances in Neural Information Processing Systems, vol. 35, pp. 24824–24837, 2022.

[11] H. Zhang, L. Zhao, and Y. Liu, Jellyfish: A Large Language Model for Data Preprocessing, IEEE Transactions on Knowledge and Data Engineering, vol. 36, no. 5, pp. 2123–2136, 2024.

[12] M. Mahdavi, Z. Abedjan, and I. F. Ilyas, Baran: Effective Error Correction via a Unified Context Representation and Transfer Learning, Proceedings of the VLDB Endowment, vol. 13, no. 12, pp. 1948–1961, 2020.

[13] Z. Miao, Y. Li, and X. Wang, Rotom: A Meta-Learned Data Augmentation Framework for Entity Matching and Data Cleaning, in Proceedings of the 2021 International Conference on Management of Data, pp. 1303–1316, 2021.

[14] T. Rekatsinas, X. Chu, and I. F. Ilyas, Holoclean: Holistic Data Repairs with Probabilistic Inference, Proceedings of the VLDB Endowment, vol. 10, no. 11, pp. 1190–1201, 2017.

[15] T. Schick, J. Dwivedi-Yu, and R. Dessi, Toolformer: Language Models Can Teach Themselves to Use Tools, Advances in Neural Information Processing Systems, vol. 36, pp. 68539–68551, 2023.

[16] N. Shinn, B. Labash, and A. Gopinath, Reflexion: An Autonomous Agent with Dynamic Memory and Self-Reflection, Journal of Artificial Intelligence Research, vol. 77, pp. 451–480, 2023.

[17] D. Chen, H. Wu, and Y. Yang, Data-Juicer: A One-Stop Data Processing System for Large Language Models, in Companion of the 2024 International Conference on Management of Data, pp. 120–134, 2024.

[18] H. Chen, Z. Wang, and L. Sun, Maybe Only 0.5% Data Is Needed: Preliminary Exploration of Low Training Data Instruction Tuning, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 8, pp. 10245–10252, 2023.

[19] L. Chen, M. Zaharia, and J. Yoon, Punica: Multi-Tenant LoRA Serving, Proceedings of Machine Learning and Systems, vol. 6, pp. 1–13, 2024.

[20] A. Chevalier, A. Wettig, and D. Chen, Adapting Language Models to Compress Contexts, Computational Linguistics, vol. 49, no. 3, pp. 701–730, 2023.

[21] S. Yao, J. Zhao, and D. Yu, ReAct: Synergizing Reasoning and Acting in Language Models, in International Conference on Learning Representations, 2023.

[22] Y. Qin, S. Liang, and Y. Feng, ToolLLM: Facilitating Large Language Models to Master 16000+ Real-World APIs, IEEE Transactions on Software Engineering, vol. 50, no. 4, pp. 1123–1137, 2024.

[23] J. Peng, Y. Tang, and H. Garcia-Molina, Self-Supervised and Interpretable Data Cleaning with Sequence Generative Adversarial Networks, " Proceedings of the VLDB Endowment, vol. 16, no. 3, pp. 433–446, 2022.