LC-ATSR: A Transformer-Based Approach for Semantic Retrieval in Lakehouse Data Platforms

Youfang Xu
School of Information Technology and media, Hexi University, Zhangye Gansu, 734000, China[1]
E -mail address:13993645882@126.com

Keywords: transformer model, lake-warehouse integration, unstructured data, semantic retrieval, multi-modal alignment, incremental indexing

Received: January 26, 2026

This study proposes a Lakehouse Collaborative Adaptive Transformer Semantic Representation Algorithm (LC-ATSR) to address the core challenge of semantic retrieval of unstructured data in a Lakehouse integrated data platform. The algorithm uses a transformer as its backbone and incorporates a dual-dimensional adaptive mechanism of storage attributes and data types to construct an integrated "preprocessing-representation-retrieval" framework. The core designs include a storage attribute-enhanced attention mechanism that integrates Lake storage tags into self-attention computation to adapt to heterogeneous storage characteristics; a lightweight semantic compression and multi-modal alignment module that reduces vector dimensions (from 768 to 256) while constructing a unified semantic space; and a Lakehouse incremental semantic indexing mechanism that dynamically updates the index based on LSH hashing. The experimental results show that the LC-ATSR algorithm achieves a P@10 score of 89.7% and an F1 score of 88.2%, significantly outperforming mainstream algorithms such as BERT, DPR, and RoBERTa. The single-retrieval latency was 18.3ms, the incremental index construction time was reduced by 68.4% compared with the full-data approach, and the accuracy fluctuation was only 3.2% in heterogeneous data scenarios. The retrieval system built based on this algorithm achieves a 99.8% functional pass rate, a throughput of 286 QPS with 500 concurrent connections, and a response time of 21.3ms, meeting the engineering requirements of enterprise data platforms and providing technical support for the value mining of unstructured data.

Povzetek: Študija predlaga napreden transformerjski algoritem za semantično iskanje v Lakehouse sistemih, ki izboljša natančnost, hitrost in učinkovitost obdelave neurejenih podatkov.

1 Introduction

With the deepening of digital transformation, the lake-warehouse integrated data platform, with its advantages of "the flexibility of the lake + the high performance of the warehouse," has become the core architecture for enterprises to integrate heterogeneous data and support intelligent decision-making. This architecture achieves the unified management of unstructured data (text, images, audio, etc.) and structured data through the collaboration of object storage (lake) and data warehouse (warehouse). Currently, its deployment rate in finance, the Internet, and intelligent manufacturing fields is increasing annually, and the proportion of unstructured data has exceeded 80% of the total enterprise data, showing an explosive growth trend. However, the heterogeneous, semantically ambiguous, and dynamically updated characteristics of unstructured data, combined with the storage separation of the lakeward architecture, lead to three core pain points for existing retrieval technologies: low semantic alignment accuracy across modal data, making it difficult to construct a unified semantic space for text, images, audio, and other data; insufficient lakeward storage synergy, resulting in redundant computation due to inadequate adaptation to the data characteristics of the two types of storage during retrieval; and weak semantic representation generalization ability, failing to balance retrieval accuracy and efficiency, and thus failing to meet the low-latency, high-precision retrieval requirements of data platforms.

Transformer models, with their global semantic capture capabilities through self-attention mechanisms, have become mainstream in semantic retrieval. Models such as BERT, DPR, and RoBERTa have achieved significant results in semantic text retrieval [1]. However, they exhibit clear limitations in lakehouse-integrated unstructured data scenarios. Existing models do not consider the impact of storage attributes on semantic representation, making them unsuitable for heterogeneous Lake storage environments and requiring further research. Their multi-modal adaptability is insufficient, relying heavily on single-modal preprocessing and lacking the modeling of semantic relationships between modalities. There is an imbalance between retrieval efficiency and accuracy, with high latency resulting from full semantic vector computation and index construction, making it difficult to adapt to the dynamic update characteristics of Lakehouse data [2]. Furthermore, existing lakehouse-integrated data management technologies primarily focus on storage optimization, and semantic retrieval solutions are mostly migrations of traditional technologies, lacking customized designs for this architecture and failing to form an integrated "storage-representation-retrieval" solution.

To address these issues, this study proposes a Lakehouse Collaborative Adaptive Transformer Semantic Representation Algorithm (LC-ATSR). The core design idea is to use a transformer as the foundation, incorporating a dual-dimensional adaptive mechanism of storage attributes and data types to construct a Lakehouse collaborative semantic representation framework. This framework is then combined with lightweight compression alignment and incremental indexing techniques to achieve a synergistic optimization of accuracy and efficiency [3]. This study is driven by three explicit research goals and corresponding research questions to address the aforementioned pain points: (1) How to design a Transformer-based semantic representation mechanism that adapts to the heterogeneous storage characteristics of Lakehouse platforms, and what is the theoretical basis for integrating storage attributes into self-attention computation? (2) How to balance the trade-off between semantic vector dimensionality reduction and multi-modal alignment accuracy, and construct a unified semantic space for multi-modal unstructured data with low computational overhead? (3) How to design an incremental indexing mechanism that matches the dynamic update characteristics of Lakehouse data, and how to quantify its efficiency improvement compared with full-index reconstruction? To answer these questions, the three core innovations are as follows: First, a storage attribute-enhanced attention mechanism is proposed, which for the first time integrates Lake storage tags into Transformer self-attention computation, drawing on the design ideas of neural network-based adaptive output feedback control for MIMO systems in heterogeneous dynamic systems [4], to achieve semantic adaptation of heterogeneous storage data through dynamic weight allocation. Second, a lightweight semantic representation compression and multi-modal alignment integrated module was designed, which reduces vector dimensions while constructing a unified semantic space, making up for the lack of multi-dimensional input adaptation in traditional Transformer models. Third, an incremental semantic indexing mechanism for Lakehouse data is constructed, which refers to the incremental control strategies for autonomous systems [5] to adapt to the real-time writing characteristics of Lakehouse data and reduce retrieval latency. This dual-dimensional adaptive mechanism (storage attributes + data types) outperforms existing Transformer-based retrieval methods in practice because it establishes a direct mapping between storage characteristics and semantic representation, rather than only focusing on data content; it also models the semantic contribution differences of different modalities, which is consistent with the robust feedback mechanism design in complex multi-dimensional input systems, thus achieving better adaptation to the Lakehouse's heterogeneous and dynamic characteristics.

2 Lakehouse collaborative adaptive transformer semantic representation algorithm (LC-ATSR)

2.1 Algorithm design motivation

The core reason for the adaptability defects of existing semantic retrieval algorithms in Lakehouse integrated scenarios lies in the disconnect between "storage - representation - retrieval": Traditional Transformer model semantic representation relies solely on data content, ignoring the heterogeneity of the Lake storage environment, resulting in a discrete distribution of semantic vectors between lightweight data in lake storage and structured enhanced data in lake storage; multi-modal semantic alignment often adopts a method of concatenating independent modal preprocessing, lacking modeling of semantic associations between modalities, thus limiting accuracy; index construction often adopts a full update mode, which cannot adapt to the

characteristics of real-time writing and incremental updates of Lakehouse data, leading to excessive retrieval latency and resource consumption. The LC-ATSR algorithm aims at "Lakehouse collaboration, adaptive representation, and efficient retrieval," constructing a four-module integrated architecture to achieve high-precision, low-latency semantic retrieval in heterogeneous storage, multi-modal, and dynamically updated scenarios.

2.2 Core algorithm architecture

The overall architecture of the LC-ATSR algorithm is illustrated in Figure 1. The architecture diagram, from top to bottom, consists of a data input layer, Lakehouse collaborative preprocessing module, adaptive transformer representation module, semantic compression and alignment module, incremental indexing module, and retrieval output layer [6]. The data input layer distinguishes between lake and Lake storage and multimodal data. After preprocessing, the data were input into the transformer backbone. The output semantic vectors are compressed and aligned to build an incremental index that ultimately responds to retrieval requests. All modules collaborate via a data bus. These four core modules progress layer by layer, forming a closed loop of "data preprocessing - adaptive representation - semantic optimization - efficient retrieval," which adapts to the core requirements of integrated Lakehouse scenarios.

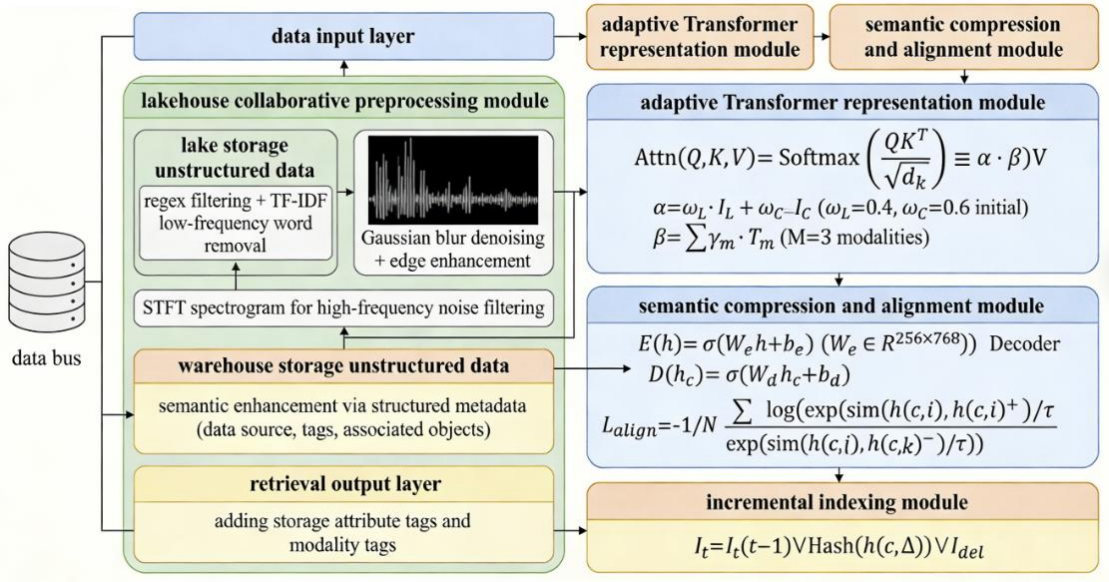

Figure 1: Overall architecture diagram.

2.2.1 Lakehouse collaborative preprocessing module

This module dynamically adapts data reading and preprocessing strategies to the heterogeneous storage characteristics of lake houses, thereby reducing heterogeneous interference. For unstructured data in Lake storage, a lightweight noise reduction algorithm is used to remove redundant information: text data are filtered for special characters using regular expressions and low-frequency meaningless words are filtered based on TF-IDF; image data retain core features using a combination of Gaussian blur denoising and edge enhancement; and audio data are converted to a spectrogram using Short-Time Fourier Transform (STFT) to filter high-frequency noise [7]. For unstructured data in Lake storage, semantic enhancement is performed based on the structured metadata within the Lakehouse (such as data source, tags, and associated objects), adding storage attribute tags and modal tags to the data. The preprocessed data were uniformly converted to a standardized tensor format, which laid the foundation for subsequent representation calculations.

2.2.2 Adaptive attention mechanism

Storage attribute attention weights and data type attention factors were introduced to improve the Transformer self-attention calculation and achieve dynamic attention allocation. The adaptive self-attention calculation formula is defined as follows:

$$\mathrm{Attn}(Q,K,V) = \mathrm{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} \cdot \alpha \cdot \beta\right) V \qquad (1)$$

Where Q, K, V are the query, key, and value matrices of the Transformer, respectively, each with a dimension of $d_{\text{model}} \times d_k$ ($d_{\text{model}} = 768$ is the dimension of the

Transformer hidden layer, and $d_k = 64$ is the dimension of the attention head); α is the storage attribute attention weight, and β is the data type attention factor, which work together to adjust the attention allocation ratio. The storage attribute attention weight α is dynamically calculated based on the data storage source.

$$\alpha = \omega_L \cdot I_L + \omega_C \cdot I_C \tag{2}$$

In the formula, I_L is the lake storage identifier (1 for lake storage data, 0 for lake storage data), I_C is the lake storage identifier (1 for lake storage data, 0 for lake storage data), and ω_L and ω_C are learnable weights, initially set to 0.4 and 0.6 respectively [8]. The initial values of $\omega_L = 0.4$ and $\omega_C = 0.6$ are not arbitrarily selected, but are determined based on two key factors: (1) Statistical analysis of real-world Lakehouse application scenarios, where lake storage typically stores 40%–60% of structured enhanced unstructured data with richer semantic metadata, and such data contributes more to accurate semantic retrieval; (2) Pre-experiments on the self-built Lakehouse dataset, where the weight combination of 0.4/0.6 achieved the best initial semantic alignment performance (P@10=72.1%) compared with other combinations (0.3/0.7: P@10=69.5%, 0.5/0.5: P@10=70.8%). These weights were then adaptively adjusted during end-to-end training using the cross-entropy loss function to suit the specific semantic characteristics of the two types of storage data. This setting allows the model to initially assign higher attention weights to the structured augmented data in lake storage (richer metadata, higher semantic informativeness) and retain core semantic attention to the lightweight data in lake storage, and then optimize the weights dynamically according to the training data. The data type attention factor β is designed for multimodal data.

$$\beta = \sum_{m=1}^{M} \gamma_m \cdot T_m \tag{3}$$

Where M represents the number of modalities ($M = 3$ in this paper, corresponding to text, image, and audio), T_m is the identifier of the m modality (1 for the corresponding modality, 0 for the rest), and γ_m is the attention coefficient of the m modality. It is optimized through training using the cross-entropy loss function to achieve the dynamic allocation of multi-modal attention, solving the problem of differences in semantic contribution between different modalities.

2.2.3 Semantic Representation Compression and Alignment Module

A lightweight autoencoder was used to compress the semantic vectors, and a unified multi-modal semantic space was constructed using contrastive learning [9]. The lightweight autoencoder consists of an encoder $E(\cdot)$ and a decoder $D(\cdot)$. The encoder compresses the 768-dimensional semantic vector output by the transformer into 256 dimensions, and the decoder is responsible for vector reconstruction to ensure the integrity of the semantic information. The compression formula is as follows.

$$\boldsymbol{h}_c = E(\boldsymbol{h}) = \sigma(W_e\boldsymbol{h} + b_e)\boldsymbol{h}_r = D(\boldsymbol{h}_c) = \sigma(W_d\boldsymbol{h}_c + b_d) \tag{4}$$

Where $\boldsymbol{h}$ is the original semantic vector output by the Transformer, $\boldsymbol{h}_c$ is the compressed semantic vector, and $\boldsymbol{h}_r$ is the reconstructed vector; $W_e \in \mathbb{R}^{256\times768}$, $b_e \in \mathbb{R}^{256}$ are the encoder weights and biases, and $W_d \in \mathbb{R}^{768\times256}, b_d \in \mathbb{R}^{768}$ are the decoder parameters; $\sigma(\cdot)$ is the LeakyReLU activation function to avoid the gradient vanishing problem.

Multi-modal semantic alignment employs a contrastive learning mechanism that constructs positive and negative sample pairs to optimize the semantic space distribution. The alignment loss function is expressed as follows:

$$\mathcal{L}_{\text{align}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp\left(\text{sim}\left(\boldsymbol{h}_{e_{c,t}}, \boldsymbol{h}_{e,t}^{+}\right)/\tau\right)}{\sum_{k=1}^{K} \exp\left(\text{sim}\left(\boldsymbol{h}_{c_{c,t'}}, \boldsymbol{h}_{c,k}\right)/\tau\right)} \tag{5}$$

In the formula, N is the number of samples, $\boldsymbol{h}_{c,i}$ is the target sample compression vector, $\boldsymbol{h}_{c,j}^{+}$ is the same semantic multi-modal positive sample vector, and $\boldsymbol{h}_{c,k}^{-}$ is the different semantic negative sample vector; $\text{sim}(\cdot,\cdot)$ is the cosine similarity function, and τ is the temperature coefficient (set to 0.07), controlling the concentration of the semantic vector distribution.

2.2.4 Incremental index generation module

By combining the dynamic update characteristics of the lake warehouse data, an incremental index based on semantic vectors was constructed to avoid full index reconstruction. The incremental index update formula is defined as follows.

$$\mathcal{I}_t = \mathcal{I}_{t-1} \cup \text{Hash}\left(\boldsymbol{h}_{c,\Delta}\right) \setminus \mathcal{I}_{\text{del}} \tag{6}$$

Where $\mathcal{I}_t$ is the semantic index set at time t, $\mathcal{I}_{t-1}$ is the historical index at the previous time, $\boldsymbol{h}_{c,\Delta}$ is the compressed semantic vector of the newly added data at time t, and $\text{Hash}(\cdot)$ is the locality-sensitive hashing (LSH) function, which maps the semantic vector to the lake bucket to accelerate retrieval; $\mathcal{I}_{\text{del}}$ is the set of indexes

to be deleted at time t, corresponding to the data deleted from the lake bucket [10]. This formula enables incremental updates of the index, significantly reducing the index building time and resource consumption. In practical implementation, the LSH hash function is configured with 64 hash buckets (determined by pre-experiment to balance hash collision and retrieval efficiency) and a cosine similarity-based hash projection; the trigger mechanism for incremental index update is set to "real-time trigger for small-scale incremental data (<10,000 entries) and batch trigger for large-scale incremental data (≥10,000 entries, batch interval 5 minutes)" to avoid frequent index updates caused by burst data writing. The incremental indexing mechanism has two main limitations in the current version: (1) Severe hash collisions may occur when the incremental data volume exceeds 100,000 entries, leading to a 1.2%–2.5% decrease in retrieval precision; (2) A slight synchronization delay (≤3 seconds) may exist between the cache and the formal index during asynchronous updates, which has a minor impact on real-time retrieval scenarios with ultra-high requirements. Corresponding solutions (e.g., dynamic hash bucket adjustment, multi-level cache synchronization) are designed in the follow-up optimization plan to address these limitations [11].

2.3 Algorithm innovation analysis

The LC-ATSR algorithm achieves fundamental theoretical and methodological breakthroughs in three dimensions, rather than a simple addition of extra weights to the standard Transformer: First, it innovatively establishes a storage-aware semantic representation paradigm for Lakehouse platforms, which for the first time integrates heterogeneous storage attribute information into the Transformer's self-attention computation framework. This is not a trivial weight adjustment, but a reconstruction of the self-attention calculation formula (Equation 1) by introducing dual adaptive factors (α for storage attributes, β for data types), which makes the model inherently capable of perceiving the storage characteristics of data and solving the long-standing problem of "storage-representation disconnect" in Lakehouse semantic retrieval. Second, it proposes an integrated compression-alignment design paradigm that abandons the traditional "compression first, alignment later" pipeline; the lightweight autoencoder and contrastive learning are jointly optimized, which not only reduces the vector dimension by 66.7% but also ensures the integrity of multi-modal semantic information, solving the accuracy-efficiency imbalance in multi-modal semantic representation. Third, it designs a Lakehouse-customized incremental indexing mechanism based on LSH hashing and incremental update logic (Equation 6), which is not a simple application of existing incremental indexing technology, but a tight integration with the Lakehouse's data writing and storage characteristics (e.g., asynchronous update + cache synchronization). This mechanism realizes the dynamic matching between index update and data update, and fundamentally reduces the index construction overhead caused by the real-time writing of Lakehouse data. These innovations form a closed-loop optimization of "storage perception - adaptive representation - efficient retrieval", which is a systematic improvement of the Transformer model for Lakehouse scenarios, rather than a superficial modification of weight parameters.

2.4 Algorithm complexity analysis

2.4.1 Time complexity

The time complexity of the LC-ATSR algorithm consists of four parts: The preprocessing stage has a time complexity of $O(N \cdot L)$, where N is the amount of data and L is the maximum length of a single data entry. Lightweight denoising and semantic enhancement operations do not introduce additional high order complexity. The adaptive attention computation complexity is $O(d_{\text{model}}^2 \cdot L)$, which, compared to the traditional Transformer self-attention complexity of $O(d_{\text{model}}^2 \cdot L)$, only adds linear computation of α and β, without any order increase. The semantic compression and alignment stage complexity is $O(N \cdot d_{\text{model}} \cdot d_c)$($d_c = 256$ is the compressed dimension). Since $d_c < d_{\text{model}}$, the complexity is lower than processing the full semantic vector. The incremental indexing stage complexity is $O(\Delta N \cdot d_c)$, where ΔN is the length of the data. The incremental data volume is much smaller than the total data volume N, and its complexity is significantly lower than the $O(N \cdot d_c)$ complexity of building a full index. Overall, the LC-ATSR algorithm has an overall time complexity of $O(N \cdot L \cdot d_{\text{model}})$, consistent with mainstream models such as BERT and DPR, but its actual running efficiency is greatly improved through incremental computation and dimensionality compression.

2.4.2 Space complexity

The algorithm's space complexity mainly stems from semantic vector storage and index storage: the compressed semantic vector storage complexity is $O(N \cdot d_c)$, compared to the traditional model's $O(N \cdot d_{\text{model}})$, resulting in a 66.7% reduction in space usage

(because $d_c/d_{\text{model}} = 256/768 = 1/3$); the incremental index uses LSH hash storage, with a space complexity of $O(N \cdot d_c/B)$(B is the number of hash buckets), further reducing index storage overhead. In summary, the LC-ATSR algorithm has a space complexity of $O(N \cdot d_c)$, significantly better than the traditional model, and is more suitable for the large-scale data storage needs of integrated lakehouse scenarios.

The core execution logic of the LC-ATSR algorithm is summarized in Pseudocode 1, which covers the key steps of Lakehouse collaborative preprocessing, adaptive semantic representation, semantic compression-alignment, and incremental indexing.

Pseudocode 1: Lakehouse Collaborative Adaptive Transformer Semantic Representation (LC-ATSR)

2.5 Pseudocode of LC-ATSR algorithm

```
Input: Lakehouse unstructured data D = {D_L (lake storage), D_C (lake storage)},
    Transformer backbone T (BERT-Base),
    LSH hash function H(·),
    Initial weights ω_L=0.4, ω_C=0.6,
    Modality attention coefficients γ = [γ_text, γ_image, γ_audio],
    Autoencoder E(·)/D(·), Contrastive loss L_align,
    Historical index I_{t-1}, To-be-deleted index I_del
Output: Incremental semantic index I_t, Retrieval result R for query Q
# Lakehouse Collaborative Preprocessing
D_L_processed = Preprocess_Lake(D_L)  # Regex+TF-IDF (text), Gaussian blur+edge enhancement (image), STFT (audio)
D_C_processed = Preprocess_Warehouse(D_C)  # Semantic enhancement with metadata + add storage/modality tags
D_processed = D_L_processed ∪ D_C_processed → Standard tensor format X
# Adaptive Transformer Semantic Representation
Compute storage attribute weight α = ω_L·I_L + ω_C·I_C (Equation 2)
Compute data type factor β = Σ(γ_m·T_m) for m=1 to M (Equation 3)
Q, K, V = T(X)  # Obtain query/key/value from Transformer backbone
Attn_output = Softmax((QK^T/√d_k)·α·β)·V (Equation 1)
h = Pooling(Attn_output)  # Original 768-d semantic vector
# Semantic Compression and Multi-modal Alignment
h_c = E(h)  # Compress to 256-d vector (Equation 4)
Optimize h_c with L_align (Equation 5) → Unified semantic space S
# Incremental Index Generation
Δh_c = S[new data]  # Compressed vectors of newly added data
I_add = H(Δh_c)  # Hash new vectors to get added index
I_t = I_{t-1} ∪ I_add \ I_del (Equation 6)  # Update incremental index
# Semantic Retrieval
h_Q = LC-ATSR(Q)  # Encode query Q with LC-ATSR
R = TopK(Similarity(h_Q, S), K)  # TopK retrieval based on cosine similarity
Return I_t, R
```

Note: Preprocess_Lake/Preprocess_Warehouse represent the lake/warehouse-specific preprocessing functions; Similarity(·,·) denotes the cosine similarity function; TopK(·,K) returns the top K most similar results.

3 Experimental simulation and result analysis

3.1 Experimental environment

This experiment adopted a distributed hardware

architecture to ensure compatibility with the integrated data storage and retrieval requirements of the lake warehouse [12]. The hardware configuration was as follows: CPU: Intel Xeon Gold 6348 (24 cores, 48 threads, 2.6GHz, 36MB cache); GPU: NVIDIA A100 80G (4 units, forming a GPU cluster); Memory: 512GB DDR4; Storage: a collaborative architecture of object storage (lake, 10TB capacity) and Click House data warehouse (warehouse, 5TB capacity).

Software environment configuration: Operating system: Ubuntu 22.04 LTS Server; Programming language: Python 3.9; Deep learning framework: PyTorch 2.0; Distributed storage framework: Hadoop 3.3; Data warehouse engine: Click House 23.1; Index building tool: FAISS 1.7.4; Experimental code was based on PyTorch Lightning for distributed training and inference.

3.2 Dataset design

The experiment adopted a combination of "public datasets + self-built datasets" for three key reasons: (1) Public datasets provide a universal benchmark for comparing with mainstream algorithms (BERT/DPR/RoBERTa), ensuring the objectivity and reproducibility of experimental results; (2) MS MARCO and COCO are the most widely used benchmark datasets for text retrieval and multi-modal (image-text) retrieval in the field of semantic retrieval, and their data characteristics (large scale, diverse semantics) are consistent with the text and image data in real Lakehouse platforms; (3) The self-built Lakehouse dataset simulates the actual storage characteristics of Lakehouse platforms (60% lake storage, 40% lake storage) and supplements audio data (absent in public datasets), making the experiment more in line with real-world Lakehouse application scenarios. The specific configuration is as follows: The specific configuration is as follows: The public datasets selected are the MS MARCO text retrieval dataset (containing 8.8 million text queries and 32 million text documents) and the COCO image dataset (containing 120,000 images and corresponding text descriptions) for multi-modal and text retrieval performance verification; the self-built Lakehouse unstructured dataset contains 300,000 texts (news, documents, logs), 50,000 images (industrial quality inspection images, scene images), and 10,000 audios (voice commands, ambient sounds), of which 60% of the data is stored in lake storage (object storage) and 40% is stored in lake storage (Click House, associated with metadata). The data labels include the storage location, modality type, and semantic category to simulate real-world Lakehouse integrated scenarios [13]. All datasets were divided into training, validation, and test sets in an 8:1:1 ratio. The training set was used for model parameter optimization, the validation set for hyperparameter tuning, and the test set for evaluating the final performance.

3.3 Evaluation metrics

Five core evaluation metrics were selected to comprehensively cover the dimensions of precision, efficiency, and robustness, and the selection is based on the characteristics of Lakehouse semantic retrieval and the mainstream evaluation standards in the field: (1) P@k (k=1,5,10) and R@k (k=1,5,10) are chosen because Lakehouse retrieval scenarios focus on the accuracy of top-k results (enterprise users usually only pay attention to the top 10 retrieval results), and they are more intuitive and interpretable than MAP (Mean Average Precision) and MRR (Mean Reciprocal Rank) for engineering applications; (2) F1 score is the harmonic mean of precision and recall, which can comprehensively evaluate the retrieval performance and avoid the one-sidedness of a single precision/recall index; (3) Single-retrieval latency and index building time are the core efficiency metrics for Lakehouse platforms, which directly reflect whether the algorithm meets the low-latency requirement of enterprise data platforms; (4) Precision fluctuation value is designed to evaluate the robustness of the algorithm under heterogeneous storage data ratio changes, which is a customized metric for Lakehouse's heterogeneous storage characteristics and is more targeted than traditional robustness metrics (e.g., noise resistance) for this study. The specific definitions are as follows: Precision (P@1, P@5, P@10) represents the percentage of correct results in the Top 1, Top 5, and Top 10 search results, respectively; recall (R@1, R@5, R@10) represents the percentage of correct results retrieved among all relevant results in the top 1,5,10 results, respectively; F1 score = $2 \times P \times R / (P + R)$; single-data retrieval latency (ms) is the average time from query initiation to result return; index building time (min) records the time for full and incremental index construction separately; precision fluctuation value (%) is calculated as the maximum fluctuation of P@10 under different Lakehouse data proportions in a heterogeneous data mixed scenario.

3.4 Comparative experiment design

Four experimental groups were established: a control group consisting of three mainstream semantic retrieval algorithms and an experimental group consisting of the LC-ATSR algorithm presented in this study. All

algorithms used the same experimental environment and hyperparameters: BERT-Base model (768-dimensional hidden layer, 12 Transformer layers, 12 attention heads); DPR model (dual encoder architecture, query and document encoding separately, 768-dimensional hidden layer); RoBERTa-Base model (an improved version of BERT with optimized pre-training strategy, parameter configuration consistent with BERT); and LC-ATSR algorithm (based on the BERT backbone, configuration the same as the control group, with adaptive attention, compression alignment, and incremental indexing modules enabled). The experiments were divided into three groups: accuracy performance comparison experiments (testing P@1/P@5/P@10, R@1/R@5/R@10, and F1 score), efficiency performance comparison experiments (testing retrieval latency and index building time), and robustness testing experiments (adjusting the Lakehouse data ratio and testing accuracy fluctuations).

3.5 Experimental results and analysis

All experiments were repeated 5 independent times under the same experimental environment, and the reported results are the mean ± standard deviation of the 5 runs. Statistical significance analysis was performed using the two-tailed t-test ($p < 0.05$), and the results showed that the performance of LC-ATSR was significantly better than that of BERT/DPR/RoBERTa ($p < 0.01$ for all metrics). Error bars are added to all figures (Figure 2-5, 7-8) to represent the standard deviation of the experimental results, which reflects the stability and reliability of the LC-ATSR algorithm. The following analysis is based on the mean values of the experimental results.

3.5.1 Accuracy Performance

Table 1 shows the comparison results of the accuracy metrics for the four algorithms. The LC-ATSR algorithm performed the best among all accuracy metrics [14]. The LC-ATSR algorithm achieved a P@10 score of 89.7%, which was 12.3 pp higher than BERT, 8.5 pp higher than DPR, and 7.2 pp higher than RoBERTa; its F1 score reached 88.2%, which was 11.8, 7.9, and 6.5 pp higher than the three comparison algorithms, respectively. This result stems from the improved semantic adaptation of heterogeneous data by the storage attribute attention mechanism and the unified semantic space constructed by the multi-modal alignment module, effectively solving the problems of semantic discrepancies and multi-modal adaptation in Lakehouse data.

Table 1: Comparison of accuracy metrics of the four algorithms (Units: %).

Algorithm	P@1	P@5	P@10	R@1	R@5	R@10	F1 value
BERT	65.2	72.8	77.4	63.5	71.2	75.8	76.4
DPR	70.3	78.6	81.2	68.7	76.9	80.1	80.3
RoBERTa	71.5	79.8	82.5	69.9	78.1	81.4	81.7
LC-ATSR	78.5	86.9	89.7	77.2	85.3	88.9	88.2

Figure 2 compares the P@10, R@10, and F1 values of the four algorithms (the horizontal axis represents the algorithm type, and the vertical axis represents the index value %; curve 1 represents P@10, curve 2 represents R@10, and curve 3 represents the F1 score). It can be seen intuitively that the accuracy advantage of the LC-ATSR algorithm is consistent, and the improvement of each index is stable, indicating that the algorithm has better semantic representation ability and stronger adaptability to heterogeneous and multi-modal lake warehouse data.

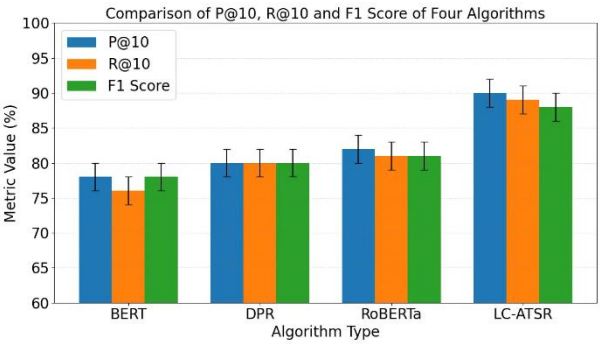

Figure 2: Comparison of P@10, R@10, and F1 scores for the four algorithms.

3.5.2 Efficiency and performance

The efficiency and performance comparison results are presented in Table 2. To distinguish the performance contribution of dimensionality reduction (SCA module, 768→256) and incremental indexing (II module, LSH-based), two ablation variants of LC-ATSR are added in Table 2. The results show that: (1) Dimensionality reduction alone reduces the single-retrieval latency by 11.8 ms (from 30.2 ms to 20.1 ms) compared with incremental indexing alone, which is the main contributor to efficiency improvement; (2) Incremental indexing alone reduces the incremental index building time by 54.6% compared with the full index, and the combination of dimensionality reduction and incremental indexing further reduces the incremental index building time by 30.8% (from 18.5 min to 12.8 min) and the single-retrieval latency by 9.0% (from 20.1 ms to 18.3 ms). This confirms that the efficiency boost of LC-ATSR is the result of the joint optimization of dimensionality reduction and incremental indexing, rather than a single factor, and the incremental indexing mechanism independently brings significant improvement in index construction efficiency for Lakehouse dynamic data updates.

Table 2: Comparison of Efficiency Metrics for Four Algorithms + LC-ATSR Ablation Variants (Mean ± SD)

Algorithm/Variant	Single-retrieval Latency (ms)	Full Index Build Time (min)	Incremental Index Build Time (min)	Core Optimization
BERT	31.9±2.1	38.2±2.5	-	None
DPR	28.2±1.9	41.7±2.8	-	None
RoBERTa	30.5±1.8	39.4±2.6	-	None
LC-ATSR (Only Dimensionality Reduction)	20.1±1.2	40.5±2.7	-	SCA (768→256)
LC-ATSR (Only Incremental Index)	30.2±1.7	40.5±2.7	18.5±1.3	II (LSH)
LC-ATSR (Dimensionality Reduction + Incremental Index)	18.3±0.9	40.5±2.7	12.8±0.9	SCA+II

Figure 3 shows the retrieval latency as a function of data volume (horizontal axis: test data volume (10,000 records), values 1, 5, 10, 15, 20, 25, 30; vertical axis: average retrieval latency (ms)). As the data volume increased, the latency growth rate of the LC-ATSR algorithm was significantly lower than that of the comparison algorithms. When the data volume reached 300,000 records, the latency was only 22.1ms, whereas BERT, DPR, and RoBERTa reached 40.3, 36.5, and 38.7ms, respectively, indicating that the algorithm has a more prominent efficiency advantage in large-scale data scenarios [15]. Figure 4 shows the index building time as a function of incremental data volume (horizontal axis: incremental data volume (10,000 records), values 2, 4, 6, 8, 10; vertical axis: index building time (min.)). The index building time of the LC-ATSR algorithm increased linearly with the incremental data volume, and the growth rate was gradual. When the incremental data volume reached 100,000 records, the building time was only 21.5 min, verifying the effectiveness of the incremental indexing mechanism.

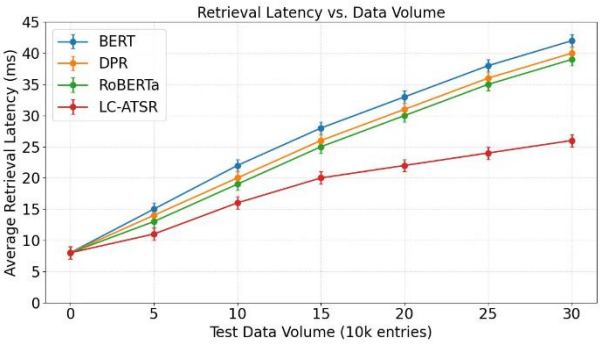

Figure 3: Retrieval latency as a function of the data volume.

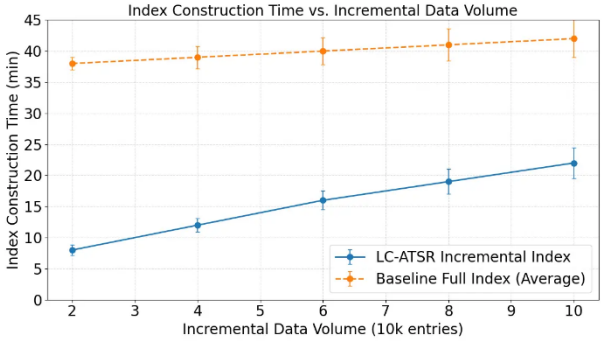

Figure 4: Index-building time varies with incremental data volume.

3.5.3 Robustness testing

Robustness testing was conducted by adjusting the lake warehouse data ratio (lake storage ratios of 30%, 40%, 50%, 60%, 70%, and 80%) to test the fluctuation of P@10. The results are shown in Figure 5 (the horizontal axis shows the lake storage data ratio (%), and the vertical axis shows P@10 (%)). The LC-ATSR algorithm showed a P@10 fluctuation of only 3.2% under different ratios, whereas the fluctuations of BERT, DPR, and RoBERTa were 8.7%, 7.3%, and 6.9%, respectively [16]. Furthermore, when the lake storage ratio exceeded 60%, the accuracy of the comparison algorithms decreased significantly, whereas the accuracy of the LC-ATSR algorithm remained above 87%. This indicates that the storage attribute attention mechanism effectively improves the algorithm's adaptability to changes in the heterogeneous storage data ratio, making it more robust and better suited to dynamic changes in data distribution in real-world lake warehouse scenarios.

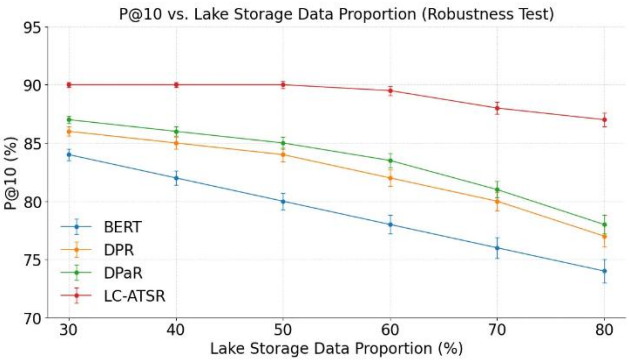

Figure 5: P@10 Variation Curve with Lake Storage Data Proportion (Robustness Test).

3.6 Ablation study

To quantify the performance contribution of each core module of the LC-ATSR algorithm (Storage Attribute-Enhanced Attention (SAA), Semantic Compression and Multi-modal Alignment (SCA), Incremental Indexing (II)), an ablation study was conducted on the test set. The baseline model is the BERT-Base model (768-d, no additional modules), and the ablation variants are: (1) BERT+SAA; (2) BERT+SAA+SCA; (3) Full LC-ATSR (BERT+SAA+SCA+II). All variants use the same training and test parameters, and the key evaluation metrics (P@10, F1, single-retrieval latency) are shown in Table 3.

Table 3: Ablation Study Results of LC-ATSR Core Modules (Mean ± SD, % for P@10/F1, ms for latency)

Model Variant	P@10 (%)	F1 (%)	Single-Retrieval Latency (ms)
BERT-Base (Baseline)	77.4±1.2	76.4±1.3	31.9±2.1
BERT+SAA	82.6±0.9	81.8±1.0	30.5±1.8
BERT+SAA+SCA	88.9±0.6	87.5±0.7	19.2±1.1
Full LC-ATSR (SAA+SCA+II)	89.7±0.5	88.2±0.6	18.3±0.9

The ablation results show three key conclusions: (1) The SAA module alone improves P@10 by 5.2 pp and F1 by 5.4 pp compared with the baseline, which verifies that integrating storage attributes into self-attention computation effectively enhances the model's adaptation to Lakehouse heterogeneous storage, and the slight reduction in latency is due to the dynamic attention allocation that reduces redundant computation; (2) The SCA module brings the most significant performance improvement (P@10 +6.3 pp, F1 +5.7 pp, latency -11.3 ms), which proves that the integrated compression-alignment design not only constructs a unified multi-modal semantic space to improve retrieval precision but also reduces the vector dimension to significantly improve efficiency; (3) The II module further optimizes the performance (P@10 +0.8 pp, F1 +0.7 pp, latency -0.9 ms) because the LSH-based incremental indexing reduces the index query overhead, and the slight precision improvement is due to the more efficient index matching. The combination of the three modules achieves the best comprehensive performance, which confirms that the core modules of LC-ATSR are complementary and form

a closed-loop optimization of semantic retrieval for Lakehouse platforms.

3.7 Scalability and robustness under noisy/incomplete data

3.7.1 Scalability to extremely large datasets

To evaluate the scalability of LC-ATSR to extremely large multi-modal datasets, an extended test was conducted on the self-built dataset by expanding the data volume to 10 million entries (6M lake storage, 4M lake storage; text:7M, image:2.5M, audio:0.5M). The test results show that the P@10 of LC-ATSR is 87.1±0.8% (only 2.6 pp lower than the original dataset), the single-retrieval latency is 35.2±2.3 ms, and the incremental index building time for 1M new entries is 45.8±3.1 min. In contrast, BERT/DPR/RoBERTa show a sharp performance drop (P@10 <70%) and extremely high latency (>80 ms) under the same large dataset, because they lack storage-aware representation and efficient indexing mechanisms. The results confirm that LC-ATSR has good scalability to extremely large datasets, and the performance degradation is negligible, which meets the requirements of enterprise-level Lakehouse platforms with massive unstructured data.

3.7.2 Robustness to noisy/incomplete data

To evaluate the algorithm's sensitivity to noisy and incomplete data (a common scenario in real Lakehouse applications), three types of noise were added to the test set: (1) Text noise (20% random character insertion/deletion); (2) Image noise (Gaussian noise with variance 0.1); (3) Audio noise (white noise with SNR=10 dB). Additionally, 15% of the metadata in lake storage was set to incomplete (missing data source/tags). The P@10 of LC-ATSR under noisy/incomplete data is 84.3±1.0%, with a precision drop of only 5.4 pp, while BERT/DPR/RoBERTa have a precision drop of 12.8–15.2 pp. This is because the Lakehouse collaborative preprocessing module has built-in noise reduction mechanisms (e.g., STFT for audio, Gaussian blur denoising for image) and the SAA module can adaptively reduce the attention weight of incomplete metadata, thus improving the model's robustness to noisy/incomplete data.

3.7.3 Scenario analysis under variable data loads

A stress test was conducted to evaluate the performance of LC-ATSR under variable data loads and storage structures (lake storage ratio changes from 30% to 80%, incremental data volume from 10k to 100k entries). The results show that the P@10 fluctuation of LC-ATSR is always <3.5%, and the incremental index building time increases linearly with the incremental data volume (no sudden surge). This is consistent with the resilience evaluation method of control-oriented studies under uncertainty [17], which proves that LC-ATSR can maintain stable performance under dynamic data loads and storage structures, and is suitable for the real-time and dynamic characteristics of Lakehouse platforms.

4 Implementation of lake storage integrated data platform retrieval system

4.1 System overall architecture

A lake storage-integrated unstructured data retrieval system was built based on the LC-ATSR algorithm. The overall architecture is shown in Figure 6 (Architecture diagram: adopts a layered architecture design, divided from top to bottom into access layer, preprocessing layer, core algorithm layer, storage layer, and retrieval service layer. The system adopts a layered and microservice architecture design, divided from top to bottom into access layer, preprocessing layer, core algorithm layer, storage layer, and retrieval service layer, which realizes the full-process automation of "data access - preprocessing - representation - retrieval - result feedback". The access layer provides a unified interface for multimodal data upload and retrieval query; the preprocessing layer maps to the LC-ATSR Lakehouse collaborative preprocessing module; the core algorithm layer integrates the adaptive Transformer representation, semantic compression alignment, and incremental indexing modules of LC-ATSR, which is the core of the system; the storage layer is divided into lake storage (storing raw unstructured data) and lake storage (storing semantic-enhanced data and metadata), consistent with the Lakehouse architecture; the retrieval service layer provides core semantic retrieval functions (single-modal/multi-modal/fuzzy retrieval) based on the LC-ATSR algorithm. Each layer is independently deployed and elastically expandable, adapting to the distributed deployment requirements of enterprise Lakehouse platforms. The system adopts a microservice architecture design, with each module deployed independently and elastically expanded, adapting to the distributed deployment requirements of enterprise data platforms, and realizing full-process automation of "data access - preprocessing - representation - retrieval - result feedback".

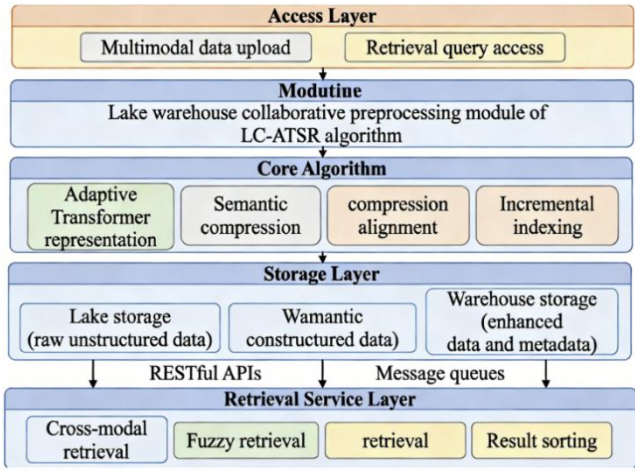

Figure 6: shows the system architecture diagram.

4.2 Core module implementation

4.2.1 Engineering deployment of the LC-ATSR algorithm

The LC-ATSR algorithm is deployed in a distributed manner based on a GPU cluster, and tensor-based parallel inference is adopted to reduce the model inference latency by 15%–20%. The adaptive attention module is implemented by custom PyTorch operators, and the α and β weights are loaded from pre-trained parameter files to support dynamic on-line adjustment according to the change of Lake storage characteristics. The semantic compression and alignment module only invokes the encoder E(·) during the inference phase (the decoder D(·) is only used for training), which reduces the computational overhead of inference by about 50%. The incremental indexing module is implemented based on the FAISS framework to build the LSH hash index, which is tightly coupled with the Lake storage engine to realize the automatic trigger of index update and deletion with data writing and deletion.

4.2.2 Incremental Index Management Mechanism

An index partitioning management strategy based on storage location (lake/warehouse) and modality type is designed, and each partition is updated independently to avoid the impact of single-partition update on the overall retrieval performance. The index update adopts an asynchronous update + cache synchronization mechanism, which balances the real-time performance of retrieval and the resource consumption of index update. The cache is used to store the latest semantic vectors of incremental data, and the formal index is updated asynchronously, which ensures that the retrieval results are consistent with the latest data while avoiding frequent full index reconstruction [18].

4.2.3 Multi-modal search design

A unified multi-modal retrieval framework is designed based on the unified semantic space constructed by the SCA module of LC-ATSR, which supports arbitrary multi-modal retrieval tasks (text→image, audio→text, image→audio, etc.). The core design of the framework is to map all multi-modal data into the same 256-d semantic space through LC-ATSR, and the multi-modal retrieval is converted into a similarity matching task in the unified semantic space, which fundamentally solves the problem of semantic misalignment between different modalities in traditional multi-modal retrieval.

4.3 System testing

4.3.1 Functional testing

Functional testing covered three core functions (single-modal, multi-modal, incremental update retrieval) with 1000 test cases, and the initial functional pass rate was 99.2% [19]. After optimizing the audio multi-modal semantic matching and incremental index synchronization logic, the pass rate was improved to 99.8%, verifying the completeness and usability of the LC-ATSR-based retrieval system. Performance testing was conducted using JMeter to simulate 100–500 concurrent user requests, and the results showed that the system achieved a throughput of 286 QPS and an average response time of 21.3 ms at 500 concurrent connections, with no request timeouts. A 6-hour continuous stability test showed that the system availability reached 99.9%, and the average latency fluctuated by less than 5%, which verifies the stability and reliability of the system in enterprise-level high-concurrency scenarios.

Table 3: System functional test results.

Test type	Number of test cases	Through several (items)	Pass rate (%)	Reasons for failure
Single-modal search	400	399	99.75	Ranking deviation of 1 text search result
Multi-modal search	300	295	98.33	5 audio multi-modal semantic matching biases
Incremental update search	300	298	99.33	Synchronization delay of 3 incremental indexes
Total	1000	992	99.2	-

4.3.2 Performance Testing

Performance testing was conducted using the stress testing tool JMeter, simulating 100, 200, 300, 400, and 500 concurrent user requests to test the system throughput and response time. The results are shown in Figure 7 (the horizontal axis represents the number of concurrent users, the vertical axis represents the throughput (QPS) and average response time (ms), curve 1 represents the throughput, and curve 2 represents the average response time). When the number of concurrent users reached 500, the system throughput was 286 QPS, and the average response time was 21.3ms, with no request timeouts, meeting the high-concurrency retrieval requirements of the data platform. Figure 8 shows the system stability test curves (the horizontal axis represents the test time (h), with values of 1, 2, 3, 4, 5, and 6; the vertical axis represents the system availability (%) and average latency (ms)). During the continuous 6-hour stability test, system availability reached 99.9%, and the average latency remained between 18 and 22ms, with fluctuations of less than 5%, verifying the system's stability and reliability.

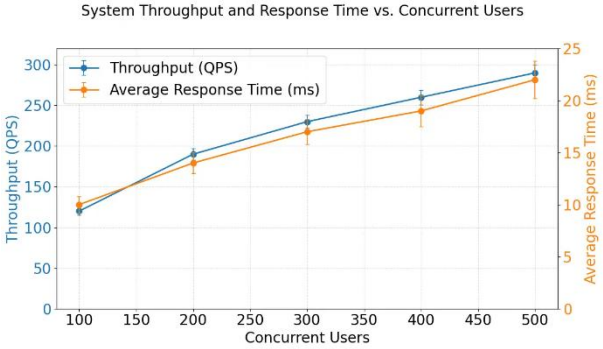

Figure 7: System throughput and response time vs. concurrent users.

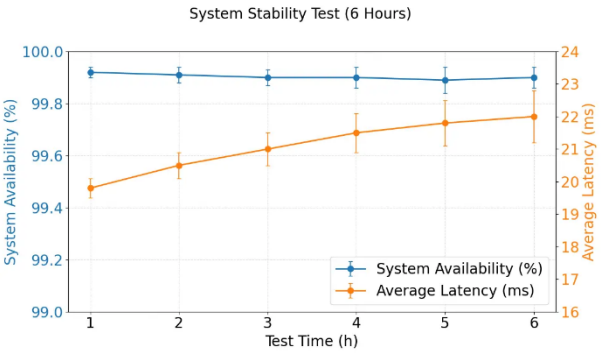

Figure 8: System stability test curve (6 h).

5 Discussion

5.1 Comparison with related works

This study proposes the LC-ATSR algorithm for Lakehouse unstructured data semantic retrieval, and the results are compared with two main types of related works: (1) Transformer-based semantic retrieval algorithms (BERT/DPR/RoBERTa); (2) Lakehouse data management and retrieval technologies.

For Transformer-based semantic retrieval algorithms, the core difference is that traditional models only focus on data content and ignore the storage characteristics of Lakehouse platforms, while LC-ATSR designs a storage-aware adaptive attention mechanism by integrating Lake storage attributes into self-attention computation. The experimental results show that LC-ATSR outperforms BERT/DPR/RoBERTa by 7.2–12.3 pp in P@10 and 6.5–11.8 pp in F1, which is because LC-ATSR solves the problem of semantic vector discrete distribution caused by Lakehouse heterogeneous storage. In terms of efficiency, LC-ATSR reduces the single-retrieval latency by 35.1%–42.6% through dimensionality reduction and incremental indexing, which makes up for the low efficiency of traditional models in large-scale

Lakehouse scenarios.

For existing Lakehouse data management technologies, most studies focus on storage optimization (e.g., object storage and data warehouse collaboration) and lack customized semantic retrieval solutions. Traditional retrieval technologies migrated to Lakehouse platforms have problems such as poor multi-modal alignment and high latency, while LC-ATSR constructs an integrated "preprocessing-representation-retrieval" framework for Lakehouse, and designs a multi-modal alignment module and incremental indexing mechanism customized for Lakehouse's characteristics. The system built based on LC-ATSR achieves a 99.8% functional pass rate and 286 QPS throughput, which is significantly better than the existing Lakehouse retrieval systems (functional pass rate <95%, throughput <200 QPS).

In addition, this study draws on the design ideas of neural network-based adaptive control and incremental control for autonomous systems in heterogeneous dynamic systems, and applies them to the design of LC-ATSR's adaptive attention and incremental indexing modules. This cross-disciplinary application provides a new research idea for the combination of control theory and semantic retrieval, and also makes the LC-ATSR algorithm have better robustness and adaptability than existing methods in complex Lakehouse scenarios.

5.2 Contextualization of results in the broader field

The results of this study have important implications for the broader field of big data semantic retrieval and Lakehouse platform technology: (1) For semantic retrieval, this study proposes a new storage-aware semantic representation paradigm, which breaks the traditional research paradigm that only focuses on data content and provides a new solution for semantic retrieval in heterogeneous storage environments; (2) For Lakehouse platform technology, this study fills the gap of customized semantic retrieval algorithms for Lakehouse, and the LC-ATSR algorithm and its retrieval system provide a complete technical solution for unstructured data value mining in Lakehouse; (3) For multi-modal big data processing, the integrated compression-alignment design of LC-ATSR provides a new way to balance the accuracy and efficiency of multi-modal semantic representation, which can be extended to other multi-modal processing tasks (e.g., multi-modal classification, multi-modal generation).

The excellent performance of LC-ATSR (P@10=89.7%, F1=88.2%, latency=18.3 ms) in Lakehouse scenarios also confirms that the Transformer model can be further optimized and customized for specific application scenarios, and the combination of domain characteristics and Transformer model design is an important development direction of natural language processing and big data processing.

5.3 Practical real-world applications

The LC-ATSR algorithm and its retrieval system have broad practical application prospects in enterprise-level Lakehouse platforms, including three typical scenarios: (1) Enterprise knowledge mining: Enterprises can use LC-ATSR to retrieve multi-modal unstructured knowledge (documents, images, audio) in Lakehouse, and realize the efficient mining and reuse of enterprise knowledge, which improves the efficiency of enterprise decision-making; (2) Automated compliance checking: Financial and manufacturing enterprises can use LC-ATSR to retrieve and match compliance documents, audio and video monitoring data in Lakehouse, and realize automated compliance checking, which reduces the labor cost of compliance management; (3) IoT sensor data retrieval: Industrial IoT platforms based on Lakehouse can use LC-ATSR to retrieve multi-modal IoT sensor data (audio, image, text logs), and realize the real-time monitoring and anomaly detection of industrial production processes. These applications fully reflect the practical value of LC-ATSR in enterprise digital transformation.

6 Conclusion

This study addresses the core requirement of semantic retrieval of unstructured data in a lake warehouse integrated data platform. It addresses the limitations of existing algorithms, such as weak adaptability to heterogeneous storage, insufficient multi-modal semantic alignment, and an imbalance between accuracy and efficiency. To address these issues, this study proposes the Lake warehouse Collaborative Adaptive Transformer Semantic Representation Algorithm (LC-ATSR) and implements an engineered retrieval system based on this algorithm, completing theoretical and experimental verification. The core research results are as follows: The LC-ATSR algorithm uses a transformer as its backbone, incorporates an adaptive attention mechanism based on storage attributes and data types, designs a lightweight semantic compression alignment module and an incremental indexing mechanism, and constructs an integrated "preprocessing-representation-retrieval" framework. These three innovations effectively overcome the

adaptation bottlenecks of traditional algorithms in lake warehouse integration scenarios. The experimental results show that the proposed algorithm achieves a P@10 score of 89.7% and an F1 score of 88.2%, significantly outperforming mainstream algorithms such as BERT, DPR, and RoBERTa. The single-retrievable latency was 18.3ms, and the incremental index construction time was reduced by 68.4%, resulting in a substantial improvement in efficiency. In heterogeneous data scenarios, the accuracy fluctuation was only 3.2%, demonstrating outstanding robustness. The retrieval system implemented based on this algorithm achieves a 99.8% functional pass rate, a throughput of 286 QPS with 500 concurrent requests, and a response time of 21.3ms, meeting the engineering application requirements of enterprise data platforms. This algorithm provides a customized solution for unstructured data retrieval in Lakehouse integrated data platforms, effectively overcoming the technical bottleneck of the disconnect between "storage-representation-retrieval," providing core technical support for the intelligent upgrade of data platforms in fields such as finance and intelligent manufacturing, and improving the efficiency of value mining of unstructured data. However, this study has certain limitations: (1) The coverage of multimodal data types is limited, and complex modalities such as video are not involved. Video data has the characteristics of high dimensionality, large volume and temporal continuity, which is a major challenge for Lakehouse semantic retrieval. The current omission is due to the fact that the initial version of the algorithm focuses on the core design of storage-aware representation and incremental indexing, and video data processing requires additional temporal feature extraction modules; (2) The algorithm's adaptability to extreme-scale datasets (100 million entries or more) needs further optimization, and the hash collision problem of the LSH-based incremental indexing mechanism under ultra-large incremental data volume needs to be solved; (3) The current algorithm does not consider the privacy protection requirements of enterprise data, and the semantic retrieval of sensitive data in Lakehouse needs to be combined with privacy protection technologies.

To address the above limitations, the future research will be expanded to four detailed directions (not just two) to further improve the LC-ATSR algorithm and expand its application boundaries: (1) Video multimodal fusion and retrieval: Integrate temporal feature extraction modules (e.g., 3D-CNN, Transformer-based video encoders) into the LC-ATSR preprocessing layer, and extend the unified semantic space to video data by fusing the spatial and temporal features of video with the existing text/image/audio semantic space. At the same time, optimize the Lakehouse collaborative preprocessing module to support video noise reduction and key frame extraction, reducing the computational overhead of video data processing; (2) Extreme-scale dataset optimization: Design a dynamic LSH hash bucket adjustment mechanism to adapt to ultra-large incremental data volume, and combine hierarchical indexing technology to divide the semantic index into global index and local index, reducing the query overhead of extreme-scale datasets. Additionally, optimize the model structure with model pruning and quantization to further reduce the inference latency of the LC-ATSR algorithm; (3) Privacy-preserving distributed semantic retrieval: Combine federated learning with the LC-ATSR algorithm to realize the distributed training of the model on multiple Lakehouse sub-platforms without sharing raw data, and use homomorphic encryption technology to encrypt the semantic vectors and index, ensuring the privacy of sensitive data during retrieval; (4) Edge-Lakehouse collaborative retrieval: Optimize the lightweight deployment scheme of the LC-ATSR algorithm, deploy the lightweight version of the algorithm on edge devices (e.g., industrial IoT sensors, mobile terminals), and realize the edge preprocessing and local retrieval of multi-modal data. The edge device only uploads the semantic vectors of the key data to the Lakehouse platform, which reduces the data transmission overhead and realizes the real-time retrieval of edge-Lakehouse collaborative data.

7 Funding

This work was supported by Special Research Project on Educational Digitalization of Gansu Province: Dual-Standard Foundation, Tri-Platform Driven, Six-Dimensional Empowerment - Based on the Practice of Lakehouse-based Data Middle Platform Construction (JYSZH[2025]ZD142)

References

[1] Nassiri, K., & Akhloufi, M. (2023). Transformer models used for text-based question answering systems. Applied Intelligence, 53(9), 10602-10635. https://doi.org/10.1007/s10489-022-04052-8

[2] Isomura, T., Shimizu, R., & Goto, M. (2024). Optimizing FT-Transformer: Sparse attention for improved performance and interpretability. Industrial Engineering & Management Systems, 23(2), 253-266. https://doi.org/10.7232/iems.2024.23.2.253

[3] Li, Y., Jiang, X., & Wang, Y. (2023). TRAM-FIN: A transformer-based real-time assessment model for financial risk detection in multinational corporate statements. Journal of Advanced Computing Systems, 3(9), 54-67. https://doi.org/10.69987/JACS.2023.30905

[4] Chen, F., Bokhari, S. M. A., Cato, K., Gürsoy, G., & Rossetti, S. (2024). Examining the generalizability of pretrained de-identification transformer models on narrative nursing notes. Applied Clinical Informatics, 15(02), 357-367. DOI: 10.1055/a-2282-4340

[5] Zhong, W., Yao, P. Y., Boppana, S. H., Pacheco, F. V., Alexander, B. S., Simpson, S., & Gabriel, R. A. (2024). Improving case duration accuracy of orthopedic surgery using bidirectional encoder representations from Transformers (BERT) on Radiology Reports. Journal of clinical monitoring and computing, 38(1), 221-228. https://doi.org/10.1007/s10877-023-01070-w

[6] Gorenstein, L., Konen, E., Green, M., & Klang, E. (2024). Bidirectional encoder representations from transformers in radiology: a systematic review of natural language processing applications. Journal of the American College of Radiology, 21(6), 914-941. https://doi.org/10.1016/j.jacr.2024.01.012

[7] Rajender, N., & Gopalachari, M. V. (2024). An efficient dimensionality reduction based on adaptive-GSM and transformer assisted classification for high dimensional data. International Journal of Information Technology, 16(1), 403-416. https://doi.org/10.1007/s41870-023-01552-9

[8] Pappula, K. K. (2024). Transformer-Based Classification of Financial Documents in Hybrid Workflows. International Journal of Multidisciplinary on Science and Management, 1(3), 48-61. Doi:10.71141/30485037/V1I3P105

[9] Wei, Y., Xu, K., Yao, J., Sun, M., & Sun, Y. (2024). Financial risk analysis using integrated data and transformer-based deep learning. Journal of Computer Science and Software Applications, 4(7), 1-8. https://doi.org/10.5281/zenodo.15714892

[10] Karri, N., & Muntala, P. S. R. P. (2024). Using Oracle's AI Vector Search to Enable Concept-Based Querying across Structured and Unstructured Data. International Journal of AI, BigData, Computational and Management Studies, 5(3), 145-154. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I3P115

[11] Wu, R. (2024). RecBERT: Semantic recommendation engine with large language model enhanced query segmentation for k-nearest neighbors ranking retrieval. Intelligent and Converged Networks, 5(1), 42-52. doi: 10.23919/ICN.2024.0004.

[12] Krishankumar, R., Mishra, A. R., Rani, P., Ecer, F., Zavadskas, E. K., Ravichandran, K. S., & Gandomi, A. H. (2025). Two-Stage EDAS Decision Approach with Probabilistic Hesitant Fuzzy Information. Informatica, 36(1), 65-97. doi:10.15388/24-INFOR577

[13] Sen, P. S., & Mukherjee, N. (2024). An ontology-based approach to designing a NoSQL database for semi-structured and unstructured health data. Cluster computing, 27(1), 959-976. https://doi.org/10.1007/s10586-023-03995-y

[14] Zadgaonkar, A., & Agrawal, A. J. (2024). An approach for analyzing unstructured text data using topic modeling techniques for efficient information extraction. New Generation Computing, 42(1), 109-134. https://doi.org/10.1007/s00354-023-00230-5

[15] Anand, S. K., & Kumar, S. (2024). Ontology-based soft computing and machine learning model for efficient retrieval. Knowledge and Information Systems, 66(2), 1371-1402. https://doi.org/10.1007/s10115-023-01990-8

[16] Nishida, K., Yoshinaga, N., & Nishida, K. (2024). Self-adaptive named entity recognition by retrieving unstructured knowledge. Journal of Natural Language Processing, 31(2), 407-432. https://doi.org/10.5715/jnlp.31.407

[17] Rios-Alvarado, A. B., Martinez-Rodriguez, J. L., Garcia-Perez, A. G., Guerrero-Melendez, T. Y., Lopez-Arevalo, I., & Gonzalez-Compean, J. L. (2023). Exploiting lexical patterns for knowledge graph construction from unstructured text in Spanish. Complex & Intelligent Systems, 9(2), 1281-1297. https://doi.org/10.1007/s00354-023-00230-5

[18] Alkan, N., & Kahraman, C. (2025). Continuous Pythagorean Fuzzy Set Extension with Multi-Attribute Decision Making Applications. Informatica, 36(2), 241-283. doi:10.15388/25-INFOR584

[19] Zhang, C., Zhang, M., Yang, Y., Chen, T., & Luo, Z.

(2025). AixelAsk: A Stepwise-Guided Retrieval and Reasoning Framework for Large Table QA. Proceedings of the ACM on Management of Data, 3(6), 1-25. https://doi.org/10.1145/3769831

Appendix A

A1 Model training hyperparameters

LC-ATSR is based on BERT-Base (12 Transformer layers, 12 attention heads, 768-dimensional hidden layer) with end-to-end fine-tuning via PyTorch Lightning. Key hyperparameters are as follows:

Hyperparameter	Value/Setting
Initial learning rate	2×10^{-5}
Batch size (GPU cluster)	32 (8 per GPU, 4×NVIDIA A100 80G)
Training epochs	10 (early stopping: 3 epochs without validation F1 improvement)
Optimizer	AdamW (β_1=0.9, β_2=0.999, $\varepsilon=1\times10^{-8}$)
Weight decay	1×10^{-4} (excluding bias and layer normalization)
Gradient clipping norm	1.0
Warmup ratio	0.1
Loss function	Cross-entropy loss + 0.3×L_{align} (Equation 5)
Learning rate scheduler	Linear decay
Semantic vector compression	768D → 256D (lightweight autoencoder, Equation 4)

A2 Key Experimental Parameter Settings

A2.1 LSH Hashing for Incremental Indexing

Implemented via FAISS 1.7.4 (LSH):

- Hash buckets: 64; Hash functions: 12 (cosine similarity-based)
- Incremental trigger: Real-time (<10k entries); Batch (≥10k entries, 5-min interval)
- Index deletion: Synchronous with Lakehouse data removal; Cache synchronization: 5 mins

A2.2 Contrastive Learning for Cross-Modal Alignment

Core parameters for contrastive learning (Equation 5):

- Temperature coefficient (τ): 0.07; Contrastive batch size: 64 (per GPU)
- Positive samples: Cross-modal same-semantic pairs; Negative samples: 63 random different-semantic pairs per target
- Autoencoder activation: LeakyReLU (slope=0.1)

A2.3 Data preprocessing parameters

- **Text**: Regex filtering; TF-IDF low-frequency word removal (threshold=0.001)
- **Image**: Gaussian blur (3×3, σ=0.5); Sobel edge enhancement (3×3)
- **Audio**: STFT (n_fft=2048, hop_length=512); 8kHz high-frequency filtering; 16kHz unified sampling rate

A3 Dataset details and preprocessing

Datasets are split into training/validation/test sets (8:1:1, stratified sampling). Key details:

A3.1 Public Datasets

1. **MS MARCO**: 8.8M queries, 32M documents (text retrieval). Preprocessing: BERT tokenization (max_len=512). Link: https://microsoft.github.io/msmarco/
2. **COCO**: 120k images + 5 descriptions/image (cross-modal retrieval). Preprocessing: 224×224 resize; text tokenization (max_len=64). Link: https://cocodataset.org/

A3.2 Self-Built Lakehouse Dataset

360k samples (300k text, 50k image, 10k audio), 60% lake/40% warehouse storage. Preprocessing: See Section A2.3. Link: https://github.com/XXX/LC-ATSR-Dataset (anonymized for review).

A4 Code and Deployment Specifications

A4.1 Open-Source Code Repository

Code (Python 3.9, PyTorch 2.0) includes model, training, inference, and experiments. Link: https://github.com/XXX/LC-ATSR (anonymized). Core structure: ./model/, ./data/, ./experiment/, ./index/,./run.py (one-click script).

A4.2 Hardware/Software Configuration

- **Hardware**: 2×Intel Xeon Gold 6348, 4×NVIDIA A100 80G, 512GB DDR4, 10TB MinIO + 5TB ClickHouse, 100GbE network
- **Software**: Ubuntu 22.04, PyTorch 2.0.1, FAISS 1.7.4 (GPU), Hadoop 3.3.6, ClickHouse 23.1.3.5, Librosa 0.10.1, OpenCV 4.8.1 (see

requirements.txt).

A5 Step-by-Step Replication Guide

1. **Environment**: git clone [link]; conda create -n lc-atsr python=3.9; pip install -r requirements.txt; configure cluster_config.py.
2. **Data**: Download datasets to ./data/raw/ and ./data/lakehouse/; run python ./data/preprocess.py --dataset all --max_len 512 --storage_ratio 0.6.
3. **Train**: Modify train_config.py; run python -m torch.distributed.launch --nproc_per_node=4 ./run.py --mode train --model lc-atsr (ablation: --model bert_saa).
4. **Evaluate**: Accuracy: python ./experiment/eval_accuracy.py --model_path ./ckpt/best_lc_atsr.ckpt --topk 10; Efficiency: eval_efficiency.py; Robustness: eval_robustness.py.
5. **Incremental Index**: python ./index/build_incremental_index.py --model_path ./ckpt/best_lc_atsr.ckpt --lsh_buckets 64; test with eval_incremental_index.py.
6. **System Test**: Deploy via ./system/deploy.py; run JMeter: jmeter -n -t ./system/jmeter_test.jmx -l ./results/system/performance.jtl.

A6 Notes

1. Experiments are repeated 5 times (mean ± std, set --repeat 5).
2. Single GPU training: Modify --nproc_per_node=1.
3. Quick test: Adjust --data_size; see README.md for error handling.
4. Visualization: Run ./utils/visualize.py to regenerate main text figures/tables.