electronics

MDPI

Article

STED-Driven Semantic-Structural Similarity Learning for Semi-Structured Data with Automatic Supervision

Nan Yin [1], Jian Liu [2], Yuan Ma [1], Jinxiao Si [1], Xi Guo [1], Xiaotong Zhang [1] and Jie He [1,*]

[1] School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China; yin_nan@xs.ustb.edu.cn (N.Y.); d202510459@xs.ustb.edu.cn (Y.M.); m202521036@xs.ustb.edu.cn (J.S.); xiguo@ustb.edu.cn (X.G.); zhangxiaotong@ustb.edu.cn (X.Z.)
[2] Advanced Technology Academy, University of Science and Technology Beijing, Beijing 100083, China; jian.liu@ustb.edu.cn
* Correspondence: hejie@ustb.edu.cn

Abstract

Semi-structured documents contain both hierarchical structure and field-level semantics. This makes similarity measurement difficult for methods that focus only on structure or only on global semantic representations. To address this problem, we propose a framework for schema-level semantic–structural similarity of semi-structured documents. We first introduce Semantic Tree Edit Distance (STED), which combines semantic node matching with hierarchical tree alignment. We then generate training pairs through controlled perturbations of real JSON documents and use their STED scores as automatic supervision. Based on this supervision, STED–GNN learns an efficient approximation of STED for repeated similarity computation. Experiments on MixedJSONBench-Doc show that STED–GNN achieves a mean ARI of 0.9799 for clustering, an accuracy of 0.9967 in the primary inductive classification evaluation, and a mean Recall@76 of 0.7289 on the low-similarity evaluation. After document representations are cached, STED–GNN reduces repeated pairwise scoring time by approximately 183.47 times relative to STED. These results show that jointly modeling semantic and structural information is effective for semi-structured document similarity, while learned approximation substantially reduces the cost of repeated pairwise comparison.

Keywords: semi-structured data similarity; semantic tree edit distance; graph neural network; automatic supervision

Check for updates

Academic Editor: Esteban Tlelo-Cuautle

Received: 26 July 2026
Revised: 11 October 2026
Accepted: 15 October 2026
Published: 19 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Semi-structured data are widely used in modern data management systems because they combine flexible schemas with hierarchical organization [1,2]. JSON is one of the most common semi-structured formats and is extensively used in document databases, information integration pipelines, and scientific data processing workflows [3,4]. Measuring similarity between JSON documents is, therefore, important for tasks such as schema matching, data integration, retrieval, clustering, and repository organization [5].

Unlike plain text or tabular data, semi-structured documents encode information through both field semantics and hierarchical organization. Two JSON documents may describe the same concept while using different field names, nesting patterns, or container structures [6]. Purely structural comparison may fail to recognize semantically related fields with different names, whereas global text representations may overlook where these fields

occur in the document hierarchy. Similarity measurement for semi-structured data, therefore, requires the joint consideration of semantic correspondence and structural context.

Existing approaches address different aspects of this problem. Structure-based methods preserve hierarchical information through tree or graph edit operations, but they have limited ability to capture semantic correspondence between differently named fields [7,8]. Schema- and embedding-based methods represent attributes or documents as dense vectors, but explicit structural alignment is generally absent during similarity computation [5]. Learned similarity models provide flexible representations, yet the resulting similarity is usually defined implicitly through training supervision [9,10]. More broadly, recent studies have explored combining explicit computational formulations with neural models in other domains [11,12]. These studies suggest a general direction in which an interpretable formulation can provide supervision for an efficient learned approximation.

Figure 1 illustrates the representation heterogeneity considered in this work. The two JSON documents describe the same movie but organize their schemas differently. Semantically related fields use different names, such as *title* and *name*, or *duration* and *running time*. Their hierarchical contexts also differ: *title* and *duration* appear under *basic_info* in one document but at the top level in the other, resulting in different root-to-node paths. In addition, *cast* is represented using different container structures. These differences motivate a schema-level semantic–structural similarity measure that jointly considers field semantics and hierarchical context.

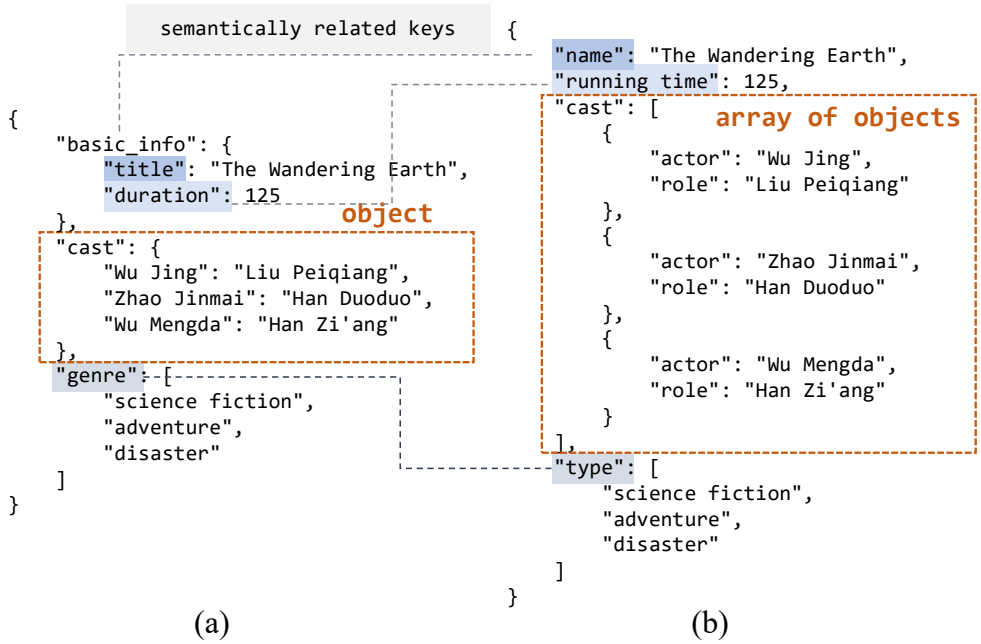


Figure 1. Two JSON documents describing the same movie: (**a**) the first JSON representation; and (**b**) the second JSON representation. The two documents differ in field names, hierarchical contexts, and container structures.

These observations lead to three main requirements. First, similarity measurement should jointly model node semantics and hierarchical structure rather than relying only on global document representations or structure-oriented edit rules. Second, manually annotating pairwise similarity for large collections of semi-structured documents is expensive and subjective, which motivates an automatic supervision mechanism. Third, explicit structural alignment can be computationally expensive when many document pairs must be compared, motivating an efficient learned approximation.

To address these requirements, we propose a framework based on Semantic Tree Edit Distance (STED) and STED–GNN. STED defines schema-level semantic–structural similarity by incorporating semantic node matching into hierarchical alignment. We further construct training pairs through controlled perturbations of real JSON documents and use their STED scores as automatic supervision. Based on this supervision, STED–GNN learns an efficient approximation of STED for repeated similarity computation. The framework is evaluated through direct approximation, clustering, classification, low-similarity detection, cross-source validation, and computational efficiency analysis.

The main contributions of this work are summarized as follows:

- We propose STED, a schema-level semantic–structural similarity measure for semi-structured documents that jointly models node semantics and hierarchical structure.
- We develop an automatic supervision strategy that constructs training pairs from real JSON documents through controlled perturbations and uses STED similarity scores as supervision.
- We develop STED–GNN as an efficient learned approximation of STED and systematically evaluate its approximation quality, downstream effectiveness, and computational efficiency under source-disjoint document and cross-source settings.

2. Related Work

Research on similarity measurement for semi-structured data can be broadly categorized into three directions: structure-based similarity methods, semantic and schema-level embedding methods, and learned similarity models. We review representative approaches in each category and discuss how our work differs from existing studies.

2.1. Structure-Based Similarity Methods

Tree edit distance (TED) is a classical approach for measuring similarity between hierarchical data structures [13,14]. It quantifies the difference between two trees through edit operations such as node insertion, deletion, and substitution. For XML documents, X-Diff provides an efficient mechanism for detecting structural changes in unordered XML trees [7], while Cobéna et al. proposed tree comparison techniques for XML version management and change detection [15]. From an algorithmic perspective, ordered tree edit distance can be computed in polynomial time, and RTED is a representative algorithm that improves robustness across different ordered tree structures while maintaining practical computational performance [16]. In contrast, computing tree edit distance for unordered trees has been shown to be NP-hard. This distinction is particularly relevant to JSON data, where arrays are ordered sequences whereas objects consist of unordered key-value pairs. To address these characteristics, JEDI extends the edit-distance paradigm to JSON documents by explicitly distinguishing ordered and unordered subtrees and defining edit operations tailored to JSON structures [8].

Structure-based methods preserve hierarchical information and provide interpretable similarity scores based on structural transformations. However, they generally rely on explicit label or value matching and have limited ability to capture semantic equivalence between differently named fields or schema contexts. As a result, documents with similar schema semantics but different structural representations may receive relatively low similarity scores. In contrast, our work incorporates semantic node matching into the edit process while preserving JSON-specific structural characteristics.

2.2. Semantic and Schema-Level Similarity Methods

To incorporate semantic information into semi-structured data analysis, a number of studies represent schemas, attributes, or entire documents using vector embed-

dings. SchemaEmbed is a representative schema-level embedding approach that combines Word2Vec and deep autoencoders to learn context-aware schema representations for JSON document clustering and similarity analysis [5]. Related studies utilize attribute names, data types, and schema structures to support schema matching and integration across heterogeneous data sources [17].

Pre-trained language models can also be used to represent semi-structured documents by linearizing JSON content into textual sequences and generating document-level embeddings. Models such as BERT [18] provide strong semantic representations, but this document-level encoding does not explicitly align hierarchical structures during similarity computation.

Despite differences in representation granularity, schema-level and document-level embedding methods primarily compare vector representations rather than performing explicit structural alignment. Consequently, hierarchical context may be only partially preserved, particularly when semantically similar fields appear at different structural positions.

In contrast, our framework incorporates semantic node representations directly into hierarchical alignment, so structural correspondence and field-level semantics are jointly considered during schema-level semantic–structural similarity computation.

2.3. Learned Similarity Models

Neural networks have been widely used to learn similarity functions directly from data. Existing approaches optimize representations for downstream tasks such as classification, retrieval, and recommendation [19,20]. Such methods can be implemented using graph neural networks [21], pre-trained language models, or large language models fine-tuned for similarity prediction [9,10]. These approaches often achieve good empirical performance and provide flexible representations for complex data.

However, learned similarity models typically require task-specific labeled or constructed supervision, and the resulting similarity function may be difficult to interpret directly. In contrast, STED-GNN is trained to approximate the explicitly defined STED similarity measure using automatically generated supervision.

3. Method

We propose a framework to compute the similarity of semi-structured documents such as JSON objects. The framework consists of three core components in Figure 2: (1) Semantic Tree Edit Distance (STED), an interpretable similarity measure that jointly captures structural and semantic information; (2) an automatic generation pipeline for supervised document pairs and (3) STED-GNN, a graph neural network designed to efficiently approximate STED similarity scores.

3.1. STED Similarity for Semi-Structured Documents

Semi-structured data represented by JSON contains both hierarchical organization and node-level textual semantics. To measure similarity between such documents, we define STED as a semantic–structural similarity measure that extends tree-edit computation with semantic node matching.

Given a JSON document, we first convert it into a semantic tree $T = (V, E)$, where each node corresponds to a structural element and edges represent parent-child relationships derived from JSON hierarchy. In this transformation, JSON objects are treated as unordered sets of child nodes, while JSON arrays are regarded as ordered sequences. This design strictly follows the inherent structural characteristics of JSON.

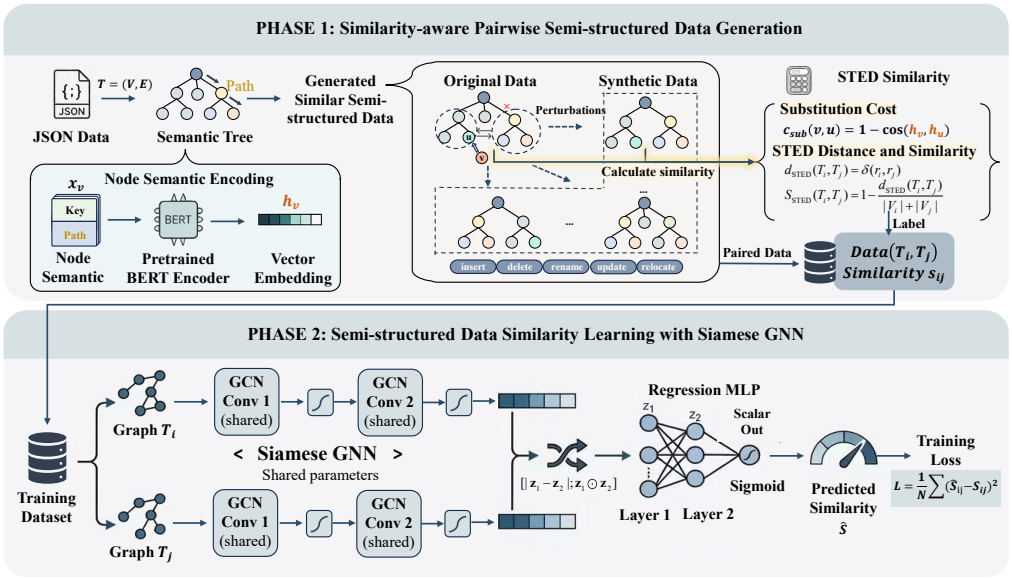


Figure 2. Overall framework for semi-structured document similarity computation. The framework consists of two phases: (1) automatic training-pair generation, which produces labeled document pairs through controlled structural and semantic perturbations and uses STED scores as automatic supervision and (2) similarity learning with STED-GNN, which learns to approximate STED for efficient similarity prediction.

To incorporate both node semantics and structural context, we construct a textual descriptor for each node v by combining its key and the corresponding path from the root node:

$$x_v = \text{concat}\big(\text{key}(v), \text{path}(v)\big). \tag{1}$$

This formulation mainly focuses on schema-level semantic and structural similarity, rather than simple matching of raw node values. Raw scalar values are not included in the current STED definition; incorporating value-aware semantics is left for future work.

We use the frozen `bert-base-multilingual-cased` encoder to obtain a 768-dimensional semantic embedding for each node:

$$\mathbf{h}_v = \text{Enc}(x_v), \tag{2}$$

where $\text{Enc}(\cdot)$ denotes the frozen multilingual BERT encoder.

For two semantic trees T_i and T_j, the substitution cost between node v and node u is defined based on the cosine similarity of their embeddings:

$$c_{\text{sub}}(v, u) = 1 - \cos(\mathbf{h}_v, \mathbf{h}_u). \tag{3}$$

The insertion and deletion costs are both set to unit cost:

$$c_{\text{ins}} = c_{\text{del}} = 1. \tag{4}$$

Because $\cos(\mathbf{h}_v, \mathbf{h}_u) \in [-1, 1]$, the substitution cost satisfies $c_{\text{sub}}(v, u) \in [0, 2]$, whose maximum equals the combined cost of deleting one node and inserting another.

STED handles object and array nodes differently. Object children are aligned using the Hungarian algorithm [22], while array elements are aligned in their original order using dynamic programming. Subtree distances are memoized to avoid repeated computation.

The accumulated edit cost is defined recursively under the JSON-specific alignment rules described above. Let r_i and r_j denote the roots of T_i and T_j. Then

$$d_{\mathrm{STED}}(T_i, T_j) = \delta(r_i, r_j), \tag{5}$$

where

$$\delta(v, u) = c_{\mathrm{sub}}(v, u) + A(C_v, C_u). \tag{6}$$

Here, $A(C_v, C_u)$ denotes the child-forest alignment cost: ordered dynamic programming is used for array children, augmented Hungarian matching is used for object children, and unmatched subtrees are charged by their full insertion or deletion costs. If the two nodes are not both arrays or both objects, all source child subtrees are deleted and all target child subtrees are inserted. The detailed recurrence is given in Appendix A. d_{STED} is used as an edit-based dissimilarity, while the normalized STED value defined below is used as the similarity score.

Finally, we normalize the accumulated edit cost to obtain the STED similarity score:

$$\mathrm{STED}(T_i, T_j) = 1 - \frac{d_{\mathrm{STED}}(T_i, T_j)}{|V_i| + |V_j|}. \tag{7}$$

The normalization yields a score in $[0, 1]$. All edit costs are non-negative, and the root nodes are kept matched. A valid transformation can always be obtained by deleting all non-root nodes of T_i, inserting all non-root nodes of T_j, and matching the two roots. Since the root substitution cost is at most 2, $0 \leq d_{\mathrm{STED}}(T_i, T_j) \leq |V_i| + |V_j|$, and, therefore, $0 \leq \mathrm{STED}(T_i, T_j) \leq 1$. A higher STED score indicates greater schema-level semantic–structural similarity.

3.2. Synthetic Training Data Generation

High-quality labeled similarity pairs of semi-structured documents are generally scarce and costly to annotate manually. To tackle this issue, we automatically generate large-scale valid document variants to build the training dataset.

Starting from original JSON documents, we generate multiple variants by performing random edit operations on their corresponding semantic trees, including node deletion, node insertion, key renaming, value modification and subtree relocation. These operations introduce controllable structural and semantic perturbations while guaranteeing valid JSON syntax and hierarchical structure.

We impose structural constraints on the maximum tree depth and total node number to ensure the generated samples are realistic. Meanwhile, semantic constraints are applied to maintain data type consistency and container compatibility. By adjusting the number and types of edit operations, the generated document pairs cover a wide range of similarity levels.

Raw scalar values are not included in the STED node descriptor. Therefore, value-only changes do not directly affect the STED score. The STED score of each generated pair is used as the supervision target for STED-GNN. Detailed generation statistics, filtering rules, and data-split audits are provided in Appendix B.

3.3. STED-GNN Model

Although STED provides interpretable semantic–structural similarity, its recursive tree alignment becomes computationally expensive when a large number of document pairs must be compared. STED-GNN, therefore, learns an efficient approximation of STED by encoding each JSON document independently into a reusable graph-level representation.

Each JSON document is converted into a bidirectional graph using the same Key + Path node representation as STED. Node texts are encoded by the frozen `bert` model, after which a shared two-layer GCN and global mean pooling produce a 512-dimensional document embedding $\mathbf{z}$.

For a document pair (d_a, d_b), the two embeddings are combined using the exchange-invariant representation

$$\mathbf{p}_{ab} = [|\mathbf{z}_a - \mathbf{z}_b| \; ; \; \mathbf{z}_a \odot \mathbf{z}_b], \tag{8}$$

which guarantees $\mathbf{p}_{ab} = \mathbf{p}_{ba}$. The resulting pair representation is passed to an MLP to predict the similarity score $\hat{s}$.

STED-GNN is trained as a regression model using STED scores as supervision, with mean squared error

$$L = \frac{1}{N} \sum_{k=1}^{N} \left(\hat{s}_k - s_k\right)^2. \tag{9}$$

The complete training and inference procedure is provided in Appendix A.

4. Experimental Setup

We evaluate the proposed framework from four perspectives: downstream performance on real JSON documents, approximation fidelity to STED, computational efficiency and scalability, and the contribution of different node representations. Correspondingly, the experiments include downstream performance comparisons, direct approximation evaluation, efficiency analysis, and ablation studies.

4.1. Datasets and Evaluation Setting

The main evaluation dataset used in this work is MixedJSONBench-Doc, which contains 120 real JSON documents from six domains, including chemistry, e-commerce, finance, games, materials, and movies, with 20 documents per domain. The documents are randomly sampled without replacement within each domain using a fixed seed of 42 and are reserved before generating the synthetic supervision data. The evaluation documents have no overlap with the source documents used for STED-GNN training according to both source-document identifiers and SHA-256 hashes of normalized document contents. For all competing methods, pairwise similarities are computed over the 120 documents and used for clustering and classification under the same evaluation protocols.

For the low-similarity evaluation, we use a fixed stratified set of 500 real JSON document pairs constructed from MixedJSONBench-Doc, covering same-source and cross-source pairs with different structural-overlap and document-size characteristics. Candidate-pair construction does not use STED, STED–GNN, JEDI, or any other evaluated similarity score. Three researchers each provide one independent rating following the same five-point ordinal rubric based on field names, field paths, hierarchical and container structure, and data types, while raw values, source/domain information, and model-generated similarity scores are excluded. Among the 500 pairs, 76 receive a score of 1 from all three annotators in their initial ratings and are used as the reference set for Recall@76.

A supplementary cross-source evaluation on independently sourced HL7 FHIR R5 JSON documents is provided in Appendix E.

4.2. Baselines

STED is included as a central reference because STED-GNN is trained to approximate its similarity scores. We compare STED-GNN with a set of representative baselines covering different technical paradigms. JEDI [8] is a classic edit-distance-based method specifically designed for JSON similarity measurement. SchemaEmbed [5] learns contextual embed-

dings to capture schema-level information. We also incorporate vanilla BERT [18] and Llama-3.2-3B [23], which convert JSON contents into plain text and calculate similarity via document-level embeddings. Furthermore, we evaluate their fine-tuned variants, namely BERT-trained and Llama-3.2-3B-finetuned. All fine-tuned models are optimized on the same synthetic dataset as STED-GNN. All methods are evaluated using the same downstream protocols for clustering, classification, and human-annotated low-similarity detection.

4.3. Evaluation Tasks and Metrics

For the clustering task, we take the similarity matrix as the affinity matrix and perform spectral clustering. The number of clusters is set to six to match the number of predefined domains in MixedJSONBench-Doc. We adopt four widely used metrics: adjusted Rand index (ARI), normalized mutual information (NMI), clustering purity and silhouette score. ARI, NMI, and purity measure agreement between the predicted clusters and the predefined domain labels, whereas the silhouette score measures cluster compactness and separation in the induced distance space. For silhouette calculation, the similarity matrix S is converted into a precomputed distance matrix as $D = 1 - S$.

For the classification task, the precomputed similarity matrix is supplied to a support vector machine (SVM) using the precomputed-kernel interface. We apply stratified five-fold cross-validation at the document level. For each fold, the classifier is fitted using only the training–training submatrix $K_{\text{train,train}}$ and the corresponding training labels, while prediction uses $K_{\text{test,train}}$; neither $K_{\text{test,test}}$ nor test labels are used during model fitting. Accuracy and Macro-F1 are reported as the averaged classification metrics across the five folds. Because a learned similarity matrix is not necessarily positive semidefinite, additional analyses of symmetry, eigenvalue spectra, and PSD correction are provided in Appendix C. An additional inductive classification evaluation using the frozen 512-dimensional STED–GNN document embeddings is also reported in Appendix C.

For low-similarity detection, each method ranks the 500 candidate pairs in ascending order of predicted similarity. Recall@76 measures the fraction of the 76 pairs receiving a score of 1 from all three annotators in their initial ratings that appear among the 76 lowest-ranked pairs predicted by each method. Spearman correlation is additionally computed between the predicted similarity scores and the final five-level reference scores over all 500 pairs. Further details are provided in Appendix F.

For methods involving stochastic training, results are obtained from five independently trained model seeds. For classification, Accuracy and Macro-F1 are first averaged over the five fixed cross-validation folds for each seed, and the reported mean and sample standard deviation are then calculated across the five seed-level averages. For clustering and low-similarity detection, each model seed produces one complete evaluation result, which is then summarized across the five seeds. Thus, the reported standard deviation reflects model-training-seed variability rather than cross-validation-fold variability. Deterministic methods are reported as single values.

4.4. Training Protocol of STED-GNN

The source collection contains 4350 real JSON documents from six domains. After reserving the 120 documents used in MixedJSONBench-Doc, the remaining 4230 source documents are split at the source-document level into 2962 training, 634 validation, and 634 test documents before any synthetic variants or document pairs are generated. Synthetic variants and document pairs are subsequently generated independently within each subset, so variants derived from the same source document cannot occur in different splits.

The resulting supervision dataset contains 15,090 document pairs, including 10,563 training pairs, 2263 validation pairs, and 2264 test pairs. Among these pairs, 9052 are

constructed from retained perturbed variants of the same source documents and 6038 are cross-document pairs. Each perturbed variant is generated using 1–8 edit requests drawn from five operation types: node deletion, node insertion, key renaming, value modification, and subtree relocation. The five edit operations are initially sampled with equal probability. After validity and structural filtering, the retained edits consist of approximately 27.6% deletion, 27.2% value modification, 19.3% subtree relocation, 13.1% insertion, and 12.9% key renaming.

Generated variants are filtered according to JSON validity, root-type consistency, nonempty keys and values, effective edits, structural constraints, and normalized-content deduplication. For every retained pair, the supervision score is calculated using STED, which serves as the training target for STED-GNN. The training, validation, and test subsets have zero overlap in source-document identifiers, normalized-content SHA-256 hashes, and generated document pairs. Detailed source-domain statistics, perturbation counts, filtering rules, structural constraints, and the resulting similarity-score distribution are reported in Appendix B.

4.5. Implementation Details

All experiments were conducted using Python 3.11.15, PyTorch 2.13.0, PyTorch Geometric 2.7.0, and scikit-learn 1.9.0. For node semantic encoding in both STED and STED-GNN, we employ `bert-base-multilingual-cased` and extract the 768-dimensional `[CLS]` embedding as node features.

The default STED-GNN is built upon a shared Siamese graph encoder with two GCN layers (hidden dimension = 512). Given two graph-level embeddings $\mathbf{z}_a$ and $\mathbf{z}_b$, the commutative pair representation $[|\mathbf{z}_a - \mathbf{z}_b|; \mathbf{z}_a \odot \mathbf{z}_b]$ is constructed and passed to the MLP prediction head, thereby enforcing $s(A, B) = s(B, A)$ by design.

The model is trained with mean squared error loss and optimized by the Adam optimizer. The learning rate is set to 3×10^{-4}, with a batch size of 4 and a total of 40 training epochs. A fixed random seed of 42 is used for single-run experiments, while stochastic results reported over repeated runs are averaged across five random seeds. Experiments are run on a Linux server equipped with an Intel Xeon Platinum 8260 CPU and a single NVIDIA Tesla P100 GPU (16 GB memory).

5. Results

5.1. Main Results

Table 1 summarizes the results on clustering and classification using MixedJSONBench-Doc, together with human-annotated low-similarity detection.

For clustering, STED and STED–GNN show numerically comparable ARI and NMI results. STED obtains an ARI of 0.9797 and an NMI of 0.9814, while STED–GNN obtains 0.9799 ± 0.0140 and 0.9827 ± 0.0110, respectively. Their reported purity values are also similar, with 0.9917 for STED and 0.9917 ± 0.0059 for STED–GNN. SchemaEmbed has the largest mean silhouette score of 0.7475 ± 0.1167, while its mean ARI, NMI, and purity are lower than the corresponding values of STED, STED–GNN, and JEDI. This difference reflects that silhouette evaluates compactness and separation in the induced distance space, whereas ARI, NMI, and purity measure agreement with the predefined domain labels.

For classification, STED achieves an Accuracy and Macro-F1 of 1.0000, while STED–GNN obtains 1.0000 ± 0.0000 for both metrics in the similarity-matrix-based evaluation. JEDI obtains an Accuracy of 0.9667 and a Macro-F1 of 0.9698. In the inductive feature-space evaluation, STED–GNN achieves an Accuracy of 0.9967 ± 0.0046 and a Macro-F1 of 0.9966 ± 0.0046.

On the low-similarity task, STED achieves a Recall@76 of 0.7368 and a Spearman correlation of 0.7312, while STED–GNN obtains 0.7289 ± 0.0072 and 0.7195 ± 0.0095, respectively. JEDI obtains a Recall@76 of 0.6316 and a Spearman correlation of 0.6966. The low-similarity reference provides an evaluation of similarity ordering without using STED scores as target labels. The numerically close results of STED and STED–GNN indicate that the learned approximation preserves much of the behavior of STED on this evaluation.

Table 1. Results on clustering, similarity-matrix-based classification, and low-similarity detection. For methods involving stochastic training, results are reported as mean ± sample standard deviation across five model seeds; deterministic methods are reported as single values.

(**a**) Clustering

Method	**ARI** ↑	**NMI** ↑	**Silh.** ↑	**Purity** ↑
STED	0.9797	0.9814	0.6586	0.9917
STED–GNN	0.9799 ± 0.0140	0.9827 ± 0.0110	0.6551 ± 0.0039	0.9917 ± 0.0059
JEDI	0.9095	0.9446	0.5300	0.9583
SchemaEmbed	0.8273 ± 0.1004	0.8596 ± 0.0554	0.7475 ± 0.1167	0.8583 ± 0.0607
BERT	0.3289	0.4923	0.3126	0.5167
BERT-trained	0.7410 ± 0.0192	0.8507 ± 0.0111	0.5386 ± 0.0713	0.7750 ± 0.0186
Llama-3.2-3B	0.4923	0.4309	0.2179	0.5250
Llama-3.2-3B-finetuned	0.8128 ± 0.0197	0.8774 ± 0.0139	0.4703 ± 0.0064	0.8667 ± 0.0220

(**b**) Classification

Method	**Acc.** ↑	**Macro-F1** ↑
STED	1.0000	1.0000
STED–GNN	1.0000 ± 0.0000	1.0000 ± 0.0000
JEDI	0.9667	0.9698
SchemaEmbed	0.8750 ± 0.0195	0.8810 ± 0.0197
BERT	0.6833	0.6734
BERT-trained	0.9000 ± 0.0118	0.9000 ± 0.0047
Llama-3.2-3B	0.7333	0.7213
Llama-3.2-3B-finetuned	0.9417 ± 0.0156	0.9385 ± 0.0161

(**c**) Low-similarity detection

Method	**Recall@76** ↑	**Spearman** ↑
STED	0.7368	0.7312
STED–GNN	0.7289 ± 0.0072	0.7195 ± 0.0095
JEDI	0.6316	0.6966
SchemaEmbed	0.1711 ± 0.0372	0.5423 ± 0.1088
BERT	0.2632	0.2534
BERT-trained	0.5684 ± 0.0195	0.6920 ± 0.0039
Llama-3.2-3B	0.4079	0.3352
Llama-3.2-3B-finetuned	0.6184 ± 0.0093	0.6205 ± 0.0346

Note: ↑ indicates that higher values are better.

Among the document-level embedding methods, fine-tuning results in higher mean Spearman correlations for both BERT and Llama-3.2-3B. BERT-trained obtains 0.6920 ± 0.0039, compared with 0.2534 for BERT, while Llama-3.2-3B-finetuned obtains 0.6205 ± 0.0346, compared with 0.3352 for Llama-3.2-3B. Their Recall@76 values also increase from 0.2632 to 0.5684 ± 0.0195 for BERT and from 0.4079 to 0.6184 ± 0.0093 for Llama-3.2-3B.

5.2. STED Approximation Fidelity

Because STED–GNN is trained using STED scores as supervision, we separately evaluate its direct approximation fidelity on the source-disjoint synthetic test set containing 2264 document pairs. Table 2 reports the prediction error and correlation between STED–GNN outputs and STED scores.

Table 2. Direct approximation fidelity of STED–GNN to STED on the source-disjoint test set. Results are reported as mean ± sample standard deviation across five model seeds.

MAE ↓	RMSE ↓	R^2 ↑	Pearson ↑	Spearman ↑
0.0076 ± 0.0005	0.0135 ± 0.0005	0.9975 ± 0.0002	0.9988 ± 0.0000	0.9954 ± 0.0003

Note: ↑ indicates that higher values are better, whereas ↓ indicates that lower values are better.

STED–GNN achieves a mean MAE of 0.0076 and a mean RMSE of 0.0135, with mean R^2, Pearson correlation, and Spearman correlation values of 0.9975, 0.9988, and 0.9954, respectively. These results indicate high fidelity to the STED supervision target on document pairs generated from source-disjoint test sources. This evaluation measures approximation fidelity and is, therefore, separated from the low-similarity evaluation in Table 1, which assesses similarity behavior using an independent reference set.

The commutative pair representation also guarantees symmetric predictions, since both $|\mathbf{z}_a - \mathbf{z}_b|$ and $\mathbf{z}_a \odot \mathbf{z}_b$ are invariant to input order. Consistent with this design, the observed symmetry error $|s(A, B) - s(B, A)|$ is zero on the evaluated test pairs.

5.3. Computational Efficiency and Scalability

We compare the computational cost of STED, JEDI, and STED–GNN when constructing pairwise similarities for the 120 documents in MixedJSONBench-Doc, corresponding to 7140 unordered document pairs in Table 3. To provide a controlled CPU-only comparison, the efficiency experiment is conducted on a MacBook Air equipped with a 10-core Apple M4 processor and 24 GB of memory. For STED and STED–GNN, runtime is decomposed into cold-start BERT loading, shared Key + Path node encoding, method-specific preparation or document encoding, and cached pairwise scoring. Both methods use the same frozen `bert-base-multilingual-cased` encoder and identical node-encoding settings. JEDI is measured using its own preprocessing and scoring pipeline.

Table 3. Efficiency comparison for 7140 document pairs under matched timing boundaries. Times are reported as mean ± SD over five independent runs.

(**a**) Representation preparation and encoding

Method	BERT Load (s)	Node Encoding (s)	Method-Specific Prep./Encoding (s)
STED	3.985 ± 0.353	273.316 ± 261.790	0.179 ± 0.346
STED–GNN	3.985 ± 0.353	273.316 ± 261.790	Graph: 0.096 ± 0.146 GCN: 0.112 ± 0.119
JEDI	–	–	0.012 ± 0.005

(**b**) Pairwise scoring and total runtime

Method	Cached Pair Scoring (s)	End-to-End Runtime (s)
STED	36.7865 ± 10.0164	314.2669 ± 261.0003
STED–GNN	0.2005 ± 0.2530	277.7505 ± 261.4123
JEDI	61.148 ± 40.886	–

Under the matched timing protocol, STED requires 36.7865 ± 10.0164 s to score all 7140 document pairs after node representations have been cached, whereas STED–GNN requires 0.2005 ± 0.2530 s after document embeddings have been cached. This corresponds to an approximately 183.47× reduction in cached pairwise scoring time. When cold-start BERT loading, shared node encoding, method-specific preparation or document encoding, and pairwise scoring are all included, the measured end-to-end runtimes are 314.2669 ± 261.0003 s for STED and 277.7505 ± 261.4123 s for STED–GNN. These results

indicate that the main computational advantage of STED–GNN lies in repeated pairwise scoring after document representations have been computed.

The scaling experiments further separate the effects of document size and the number of pairwise comparisons. Increasing document size primarily affects STED tree alignment and the one-time STED–GNN graph-construction and document-encoding stages, whereas increasing the number of documents increases the number of pairwise comparisons as $n(n-1)/2$. Detailed scaling results are reported in Appendix D.

5.4. Effect of Node Representation

We examine the contribution of Key, Path, and raw Value information to the node representation. Seven combinations are evaluated using the same source-disjoint STED approximation protocol, and MAE is used because this experiment directly measures how well each representation reproduces the STED supervision target.

As shown in Table 4, Key + Path obtains the lowest mean MAE of 0.007584, compared with 0.008170 for Path only and 0.008229 for Key only. Representations containing raw Value information produce higher approximation errors, with mean MAE increasing to 0.010293 for Path + Value, 0.011047 for Key + Path + Value, 0.011195 for Key + Value, and 0.019971 for Value only.

Table 4. Effect of different node representations on STED approximation. MAE is reported as mean ± sample standard deviation across five model seeds.

Node Representation	MAE ↓
Path only	0.008170 ± 0.000370
Key only	0.008229 ± 0.000203
Key + Path	0.007584 ± 0.000450
Path + Value	0.010293 ± 0.000548
Key + Path + Value	0.011047 ± 0.000500
Key + Value	0.011195 ± 0.000424
Value only	0.019971 ± 0.000984

Note: ↓ indicates that lower values are better.

6. Discussion

The results show that structural alignment and node semantics provide complementary information for semi-structured document similarity. JEDI performs strongly by modeling structural edit operations, while STED further incorporates semantic matching between nodes. SchemaEmbed achieves the largest mean silhouette score, but its agreement with the predefined domain labels and its performance on the low-similarity evaluation are lower. The document-level BERT and Llama baselines are less effective without explicit modeling of JSON hierarchy, although both improve after pairwise supervision. These results indicate that different methods capture different aspects of similarity, and that combining hierarchical structure with field-level semantics is useful for schema-level semantic–structural comparison.

STED and STED–GNN play different roles in the framework. STED provides an explicit similarity definition, whereas STED–GNN learns an efficient approximation for repeated pairwise comparison. The source-disjoint evaluation shows that STED–GNN closely approximates STED scores on document pairs generated from source-disjoint test sources. However, high approximation fidelity alone does not validate the underlying similarity definition. For this reason, we also evaluate the methods on the low-similarity evaluation. STED and STED–GNN obtain numerically close results on this evaluation, indicating that the learned approximation preserves much of the behavior of STED beyond the supervision data.

The current similarity definition is primarily schema-level. Node descriptors contain Key and Path information but do not include raw scalar values. Therefore, documents with similar structures and field semantics may still receive high similarity scores even when their instance values differ substantially. The node-representation experiment also shows that adding raw values does not improve approximation of the current STED target. This does not imply that values are unimportant for general document similarity; rather, value-aware comparison requires an extension of the current similarity definition.

Automatic supervision also introduces a gap between controlled perturbations and naturally occurring JSON changes. The source documents are real JSON documents, and the training, validation, and test sets are source-disjoint. However, the supervision pairs are generated from five predefined edit operations, and 69.39% of them fall in the high-similarity interval $[0.8, 1]$. Real-world changes may involve correlated field modifications, domain-specific constraints, or long-term schema evolution that are not fully represented by these perturbations. The supplementary FHIR evaluation provides additional evidence of cross-source applicability, but the lower performance of the frozen STED–GNN relative to STED also indicates that a cross-source gap remains.

The efficiency results reveal a further trade-off. After document representations are cached, STED–GNN substantially reduces the cost of repeated pairwise scoring, and this advantage becomes increasingly relevant as the number of pairwise comparisons grows. Because the STED–GNN scoring stage operates on fixed-length document embeddings, its scoring complexity does not directly depend on the number of nodes in the original JSON documents; document-size dependence is instead associated with graph construction and document encoding. In contrast, STED–GNN requires more memory than STED and JEDI because of the neural encoder and intermediate representations. STED–GNN is, therefore, more suitable for repeated or large-scale pairwise comparison, while STED remains useful when explicit edit-based computation and lower memory usage are more important.

Several directions can extend the current framework. First, value-aware similarity could incorporate numerical semantics, physical units, categorical constraints, and domain knowledge in addition to Key and Path information. Recent studies that combine explicit computational formulations with neural models provide a related methodological perspective. A similar strategy could extend STED with numerical and domain-specific constraints, while STED–GNN could learn an efficient approximation of the resulting similarity measure. Second, supervision could include naturally occurring schema revisions and independent similarity reference sets to better evaluate robustness under schema evolution and cross-source distribution shift. Third, the JSON-specific treatment of unordered objects and ordered arrays could be extended to other semi-structured formats. Finally, lightweight graph encoders, representation compression, and model distillation could reduce the memory overhead of STED–GNN for resource-constrained deployment.

7. Conclusions

This paper presents a framework for schema-level semantic–structural similarity of semi-structured documents. STED combines hierarchical tree alignment with semantic node matching to provide an edit-based similarity measure, while STED–GNN learns an efficient approximation using automatically generated supervision. Experiments on real JSON documents show that STED and STED–GNN perform competitively across clustering, classification, and low-similarity detection. The approximation analysis further shows that STED–GNN closely reproduces STED scores on source-disjoint test data, while substantially reducing repeated pairwise scoring time after document representations are cached.

The current framework also has several limitations. STED primarily models Key, Path, and hierarchical information and does not explicitly capture raw value semantics. STED–

GNN is supervised by STED and, therefore, inherits the assumptions of the underlying similarity definition. In addition, synthetic perturbations cannot fully represent naturally occurring schema evolution and cross-source variation, and the learned approximation requires higher memory consumption than STED. Future work will, therefore, focus on value-aware and numerical semantics, naturally occurring schema changes, extension to additional semi-structured formats, and more lightweight graph representations for resource-constrained deployment.

Author Contributions: Conceptualization, N.Y. and J.H.; methodology, N.Y. and J.L.; software, validation, and formal analysis, N.Y.; data curation and investigation, J.S.; writing—original draft preparation, N.Y.; writing—review and editing, J.L., Y.M., J.S., X.G. and J.H.; resources, X.Z.; supervision, J.L., X.G., X.Z. and J.H.; funding acquisition, X.Z.; project administration, X.Z. and J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work is financially supported by the Advanced Materials–National Science and Technology Major Project under Grant No. 2024ZD0607700.

Data Availability Statement: The reference implementations of STED and STED–GNN are publicly available at https://github.com/yinnan1996/STED-and-STED-GNN (accessed on 14 October 2026). The datasets used in this study were collected from publicly available sources as described in the manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Detailed Computation Procedures

Appendix A.1. STED Computation

STED converts each JSON document into a rooted semantic tree, where each node is represented by its field key and root-to-node path encoded using the frozen `bert-base-multilingual-cased` model. The detailed recursive computation is given below and summarized in Algorithm A1. With memoization, the conservative worst-case time complexity is $O(n_a n_b \Delta^3)$ and the main memoization space is $O(n_a n_b)$.

For a subtree rooted at node v, the deletion cost is defined recursively as

$$D(v) = c_{\text{del}} + \sum_{x \in C_v} D(x), \tag{A1}$$

and the insertion cost for a subtree rooted at node u is

$$I(u) = c_{\text{ins}} + \sum_{y \in C_u} I(y). \tag{A2}$$

Thus, deletion or insertion removes or adds the complete subtree without promoting its children. With unit insertion and deletion costs, $D(v)$ and $I(u)$ equal the numbers of nodes in the corresponding subtrees.

For a pair of nodes v and u, the recursive distance is

$$\delta(v, u) = c_{\text{sub}}(v, u) + A(C_v, C_u). \tag{A3}$$

For ordered child sequences $C_v = (v_1, \ldots, v_m)$ and $C_u = (u_1, \ldots, u_n)$, the alignment cost is computed by dynamic programming:

$$F_{i,j} = \min \begin{cases} F_{i-1,j} + D(v_i), \\ F_{i,j-1} + I(u_j), \\ F_{i-1,j-1} + \delta(v_i, u_j), \end{cases} \tag{A4}$$

with $F_{0,0} = 0$, $F_{i,0} = F_{i-1,0} + D(v_i)$, and $F_{0,j} = F_{0,j-1} + I(u_j)$.

For unordered child sets, an augmented Hungarian cost matrix is constructed. A real child–child assignment has cost $\delta(v_i, u_j)$, an unmatched source child has cost $D(v_i)$, an unmatched target child has cost $I(u_j)$, and dummy–dummy assignments have zero cost. The minimum assignment cost defines $A(C_v, C_u)$. If the two nodes use different child-alignment modes,

$$A(C_v, C_u) = \sum_{x \in C_v} D(x) + \sum_{y \in C_u} I(y). \tag{A5}$$

Algorithm A1 STED Similarity Computation

Require: JSON documents d_a, d_b
Ensure: Similarity score s
$T_a, T_b \leftarrow \text{BuildTrees}(d_a, d_b)$
Encode all nodes by $\mathbf{h}_v \leftarrow \text{BERT}(\text{KeyPath}(v))$
Set $c_{\text{ins}} = c_{\text{del}} = 1$ and $c_{\text{sub}}(v, u) = 1 - \cos(\mathbf{h}_v, \mathbf{h}_u)$
function DIST(v, u)
    $c \leftarrow c_{\text{sub}}(v, u)$
    if v, u are arrays **then**
        $c \leftarrow c + \text{OrderedDP}(C_v, C_u)$
    else if v, u are objects **then**
        $c \leftarrow c + \text{Hungarian}(C_v, C_u)$
    else
        $c \leftarrow c + \sum_{x \in C_v} D(x) + \sum_{y \in C_u} I(y)$
    end if
    return memoized c
end function
$\delta \leftarrow \text{Dist}(r_a, r_b)$
return $s \leftarrow 1 - \delta / (|V_a| + |V_b|)$

Appendix A.2. STED-GNN Training and Inference

STED-GNN approximates STED by encoding each JSON document with a shared two-layer GCN using frozen multilingual BERT Key + Path node representations. Global mean pooling produces a document embedding, and the exchange-invariant pair representation $[|\mathbf{z}_a - \mathbf{z}_b|; \mathbf{z}_a \odot \mathbf{z}_b]$ is passed to an MLP to predict the STED score. Document embeddings can be cached and reused for repeated pairwise inference. The complete training and inference procedure is summarized in Algorithm A2.

Algorithm A2 STED-GNN Training and Inference

Require: Training pairs $\mathcal{P}$ with STED labels s
Ensure: Trained model $f_{\theta,\phi}$
for each document d **do**
    $G_d \leftarrow \text{BuildGraph}(d)$
    $X_d \leftarrow \text{BERT}(\text{KeyPath}(G_d))$
end for
for each mini-batch (d_a, d_b, s) **do**
    $\mathbf{z}_a \leftarrow \text{MeanPool}(\text{GCN}_\theta(G_a, X_a))$
    $\mathbf{z}_b \leftarrow \text{MeanPool}(\text{GCN}_\theta(G_b, X_b))$
    $\mathbf{p} \leftarrow [|\mathbf{z}_a - \mathbf{z}_b|; \mathbf{z}_a \odot \mathbf{z}_b]$
    $\hat{s} \leftarrow \text{MLP}_\phi(\mathbf{p})$
    Update θ, ϕ by minimizing $(\hat{s} - s)^2$
end for
Cache $\mathbf{z}_i$ for each document during inference
return $f_{\theta,\phi}$

Appendix B. Synthetic Supervision Data Construction

The source collection contains 4350 real JSON documents from six domains. MixedJSONBench-Doc is first constructed by reserving 20 documents from each domain, resulting in 120 evaluation documents. The remaining 4230 source documents are split at the source-document level before any variants or document pairs are generated, as shown in Table A1. Synthetic variants are produced by applying 1–8 edit requests, where node deletion, node insertion, key renaming, value modification, and subtree relocation are initially sampled with equal probability. Generated variants are retained only if they satisfy JSON validity, root-type consistency, nonempty key/value, effective-edit, structural, and duplicate-removal constraints.

Table A1. Source-document distribution and source-disjoint data split.

Item	Number
Chemistry source documents	119
E-commerce source documents	250
Finance source documents	1000
Games source documents	1000
Materials source documents	1000
Movies source documents	981
Total source documents	4350
MixedJSONBench-Doc holdout	120
Remaining supervision sources	4230
Training sources	2962
Validation sources	634
Test sources	634
Training pairs	10,563
Validation pairs	2263
Test pairs	2264
Total supervision pairs	15,090

The supervision set contains 9052 same-source perturbation pairs and 6038 cross-document pairs. A total of 9052 unique perturbed variants are retained. Each source document contributes between one and three retained variants, with an average of approximately 2.14 variants per source. The statistics of the synthetic perturbation operations are shown in Table A2.

Table A2. Statistics of synthetic perturbation operations. Each operation has an initial sampling probability of 20%.

Operation	Attempts	Retained	Retained Proportion
Node deletion	7565	7557	27.6%
Node insertion	7631	3580	13.1%
Key renaming	7524	3530	12.9%
Value modification	7458	7436	27.2%
Subtree relocation	7502	5271	19.3%

The unequal retained frequencies result from operation-specific validity and structural constraints rather than unequal initial sampling probabilities. To limit unrealistic structural growth, generated variants satisfy

$$N_{\text{new}} \leq \lfloor 1.5 N_{\text{source}} \rfloor + 1, \qquad D_{\text{new}} \leq D_{\text{source}} + 2,$$

where N and D denote the number of tree nodes and tree depth, respectively. The largest retained document contains 334 nodes, and the maximum observed depth is 10.

The distribution of STED supervision scores is shown in Table A3.

Table A3. Distribution of STED supervision scores in the 15,090 generated document pairs.

STED Interval	Number of Pairs	Proportion
$[0.0, 0.2)$	1033	6.85%
$[0.2, 0.4)$	723	4.79%
$[0.4, 0.6)$	1410	9.34%
$[0.6, 0.8)$	1453	9.63%
$[0.8, 1.0]$	10,471	69.39%

All retained JSON documents are normalized and deduplicated using SHA-256 hashes. The training, validation, and test subsets have zero intersection in source-document identifiers and normalized-content hashes, and no generated document pair is duplicated across splits. The 120 documents in MixedJSONBench-Doc are also disjoint from all supervision sources.

We further audit overlap under the STED schema representation, which preserves hierarchical Key + Path structure while excluding raw scalar values. The training, validation, source-disjoint test, and MixedJSONBench-Doc sets contain 1545, 398, 395, and 84 distinct schema representations, respectively, with training–validation, training–test, and validation–test overlaps of 100, 95, and 38. Of the 84 MixedJSONBench-Doc schemas, 37 occur in the generated training pairs and 47 are unseen, corresponding to 72 and 48 evaluation documents. Thus, the evaluation is source-disjoint at the document level but not a fully unseen-schema setting.

Appendix C. Similarity-Matrix Diagnostics

The STED-GNN similarity matrix is symmetric by construction because the pair representation $[|\mathbf{z}_a - \mathbf{z}_b|; \mathbf{z}_a \odot \mathbf{z}_b]$ is invariant to input order. The observed symmetry error is zero on all evaluated pairs. To examine whether the learned similarity matrix also satisfies the positive-semidefinite (PSD) requirement of a standard Mercer kernel, we analyze its eigenvalue spectrum over five random seeds, as shown in Table A4. We additionally use the frozen STED–GNN document features as the primary inductive classification check and report full-matrix PSD correction separately as a transductive sensitivity analysis.

Table A4. Eigenvalue diagnostics of the STED-GNN similarity matrix across five random seeds.

Seed	Minimum Eigenvalue	Negative Eigenvalues	Negative Spectral Mass
42	−0.016037	1	0.01336%
52	0.000168	0	0
62	−0.004806	1	0.00400%
72	−0.021223	2	0.02181%
82	0.000302	0	0

Three of the five similarity matrices contain small negative eigenvalues, indicating that the learned similarity is not guaranteed to be strictly PSD. The maximum observed negative spectral mass is 0.02181%. As a transductive sensitivity analysis, we apply eigenvalue clipping to the complete 120×120 similarity matrix,

$$K_+ = Q \max(\Lambda, 0) Q^\top,$$

where $K = Q\Lambda Q^\top$ is the eigendecomposition of the complete similarity matrix. Because this transformation is computed using all 120 evaluation documents before cross-validation, the PSD-corrected result is used only to assess whether the small negative eigenvalues materially affect the classification results. Classification after PSD correction uses the same precomputed-kernel SVM protocol as the similarity-matrix-based evaluation in Table 1.

For the primary inductive check, each document is independently encoded into a frozen 512-dimensional STED–GNN representation. Within each stratified five-fold split, the `StandardScaler` and logistic-regression classifier are fitted only on the training data, with $C = 1.0$ fixed in advance. For each independently trained STED–GNN seed, Accuracy and Macro-F1 are first averaged across the five folds, and the reported mean and sample standard deviation are then calculated across the five model seeds. This evaluation achieves an Accuracy of 0.9967 ± 0.0046 and a Macro-F1 of 0.9966 ± 0.0046.

The full-matrix PSD-corrected evaluation remains at an Accuracy and Macro-F1 of 1.0000 ± 0.0000 across the five model seeds. This result is used only as a transductive sensitivity analysis and indicates that the small negative eigenvalues have little practical effect in this diagnostic setting. Table A5 summarizes the results of the inductive evaluation and the transductive PSD sensitivity analysis.

Table A5. Inductive feature-space classification and transductive PSD sensitivity analysis.

Evaluation Setting	Accuracy	Macro-F1
Full-matrix PSD correction (transductive)	1.0000 ± 0.0000	1.0000 ± 0.0000
Frozen STED–GNN document features (inductive)	0.9967 ± 0.0046	0.9966 ± 0.0046

Appendix D. Detailed Efficiency and Scalability

We further evaluate scalability with respect to both document size and the number of document pairs. All efficiency experiments use the same CPU-only environment as the main efficiency evaluation. Document-size scaling is evaluated using six documents at each size level, corresponding to 15 unordered pairs, while pair-count scaling measures the cached pairwise scoring stage after method-specific preprocessing or document encoding.

As shown in Table A6, as document size increases from approximately 16 to 256 nodes, the median pairwise scoring time of STED increases from 2.8579 ms to 729.4487 ms. For STED–GNN, document size mainly affects the one-time graph-construction and document-encoding stages. Once the fixed-length document embeddings have been cached, the scoring head operates on vectors of fixed dimensionality and, therefore, does not have computational complexity that depends on the number of nodes in the original JSON document. The observed wall-clock variation in the 15-pair cached-scoring measurements is, therefore, not interpreted as node-size scaling of the scoring head. Its memory consumption is higher because the neural encoder and intermediate representations must be retained during inference.

As shown in Table A7, the total cached scoring time increases with the number of document pairs. For n documents, the number of unordered comparisons is $n(n-1)/2$; therefore, this increase reflects the growing number of pairwise comparisons rather than increasing complexity of the fixed-length STED–GNN scoring head. For 7140 pairs, STED–GNN requires approximately 0.201 s for cached scoring, compared with 36.786 s for STED and 61.148 s for JEDI.

Table A6. Scalability with increasing document size.

Nodes	Method	Prep./Encoding (s)	Pair Scoring (s)	Median Time/Pair (ms)	Peak RSS (MiB)
16	STED	0.0031	0.0434	2.8579	316.98
	JEDI	0.0003	0.0068	0.4733	317.11
	STED–GNN	0.3980	0.0067	0.2600	887.09
64	STED	0.0099	0.6628	44.2099	320.78
	JEDI	0.0008	0.1019	6.8030	318.80
	STED–GNN	1.7133	0.0101	0.6566	928.45
256	STED	0.0403	10.9694	729.4487	592.33
	JEDI	0.0028	1.6875	112.8907	501.73
	STED–GNN	7.4977	0.0319	1.9850	1016.03

Table A7. Cached pairwise scoring time with increasing numbers of document pairs under the matched timing protocol.

Number of Pairs	STED (s)	JEDI (s)	STED–GNN (s)
66	1.210	0.865	0.003
276	2.604	2.294	0.003
1128	6.887	6.714	0.013
4560	28.338	31.222	0.162
7140	36.786	61.148	0.201

Appendix E. Cross-Source Validation on FHIR

We conduct a supplementary cross-source evaluation using the official HL7 FHIR R5 JSON examples [24]. From 2824 parsable files covering 157 `resourceType` categories, 18 resource types are selected with 12 documents per type, resulting in 216 documents. The top-level `resourceType` field is removed from the input and used only as an external evaluation label. No content overlap is found between the selected FHIR documents and the data used in the main experiments. All pretrained representation models are kept frozen without retraining or fine-tuning on FHIR. Accuracy and ARI are reported for classification and clustering, respectively.

As shown in Table A8, STED achieves the highest Accuracy and ARI on the independently sourced FHIR data. The frozen STED-GNN retains part of the learned semantic–structural information without retraining, although its performance is lower than that of STED and JEDI. Since FHIR provides resource-category labels rather than independently annotated pairwise similarity or longitudinal schema-evolution records, this experiment is used as supplementary evidence of cross-source generalization.

Table A8. External validation on the HL7 FHIR R5 dataset.

Method	Accuracy	ARI
STED	0.8934	0.5554
STED-GNN	0.6165	0.3695
JEDI	0.7948	0.4490
SchemaEmbed	0.3076	0.3303
BERT	0.0693	0.0446
Llama	0.1549	0.0372

Appendix F. The Low-Similarity Evaluation Details

The low-similarity evaluation uses a fixed set of 500 document pairs constructed from the 120 documents in MixedJSONBench-Doc. The set contains 85 same-source close pairs, 15 same-source distant pairs, 79 cross-source key-overlap pairs, 120 cross-source size-matched pairs, 90 cross-source unrelated pairs, and 111 size-imbalanced pairs. The candidate set was constructed using source relationship and schema-level structural characteristics, including Key/Path overlap and document size. No STED, STED–GNN, JEDI, or other evaluated similarity score was used during candidate-pair construction.

Three researchers each provided one independent rating following the same five-point ordinal rubric. The ratings considered field names, field paths, hierarchical and container structure, and data types, while raw values, source/domain labels, and model-generated similarity scores were excluded. Scores from 1 to 5 represent almost unrelated, low similarity, moderate similarity, high similarity, and nearly identical or minor schema variation, respectively. Exact three-way agreement was observed for 277 of the 500 pairs (55.4%). The mean pairwise linearly weighted Cohen's κ was 0.7488, and ordinal Krippendorff's α was 0.7330.

For the reference score used in the Spearman evaluation, the three initial ratings were first combined by their median. Pairs for which at least two researchers indicated uncertainty were selected for second-pass review. The 37 selected pairs were jointly reassessed by the three researchers using the same rubric, and the resulting consensus score replaced the initial median. Spearman correlation is calculated against the resulting final five-level reference score, with average ranks assigned to tied values. The three initial ratings and final reference scores for all 500 pairs are publicly available in the project repository.

Recall@76 is based on the 76 pairs that received a score of 1 in all three initial ratings. Let $\mathcal{R}_{76}^{\text{ref}}$ denote these 76 pairs and $\mathcal{R}_{76}^{\text{pred}}$ denote the 76 pairs assigned the lowest predicted similarities by a method. Recall@76 is defined as

$$\text{Recall@76} = \frac{\left|\mathcal{R}_{76}^{\text{pred}} \cap \mathcal{R}_{76}^{\text{ref}}\right|}{76}. \tag{A6}$$

If predicted similarities are tied at the cutoff, `pair_id` is used as a deterministic secondary ordering key. For methods involving stochastic training, the metric is computed independently for each model seed and then summarized across seeds. Table A9 reports the corresponding Hits@76 values for each evaluation run.

Table A9. Hit counts for the low-similarity evaluation. Hits@76 denotes the number of reference low-similarity pairs retrieved among the 76 lowest-ranked predictions.

Method	Seed 42	Seed 52	Seed 62	Seed 72	Seed 82	Mean Hits@76	Recall@76
STED	56	–	–	–	–	56	0.7368
JEDI	48	–	–	–	–	48	0.6316
BERT	20	–	–	–	–	20	0.2632
Llama-3.2-3B	31	–	–	–	–	31	0.4079
STED–GNN	56	56	55	55	55	55.4	0.7289 ± 0.0072
SchemaEmbed	11	11	15	11	17	13.0	0.1711 ± 0.0372
BERT-trained	43	43	41	45	44	43.2	0.5684 ± 0.0195
Llama-3.2-3B-finetuned	47	47	46	48	47	47.0	0.6184 ± 0.0093

To examine the effect of the second-pass review on the correlation evaluation, we additionally compute Spearman correlation using the initial median of the three ratings before second-pass review. Table A10 reports the results under both reference-score definitions.

Table A10. Spearman correlation using the final reference scores and the initial median scores before second-pass review.

Method	Final Reference	Initial Median
STED	0.7312	0.8351
STED–GNN	0.7195 ± 0.0095	0.7968 ± 0.0447
JEDI	0.6966	0.8216
SchemaEmbed	0.5423 ± 0.1088	0.5764 ± 0.0952
BERT	0.2534	0.2663
BERT-trained	0.6920 ± 0.0039	0.8177 ± 0.0038
Llama-3.2-3B	0.3352	0.3483
Llama-3.2-3B-finetuned	0.6205 ± 0.0346	0.8009 ± 0.0333

References

1. Durner, D.; Leis, V.; Neumann, T. JSON tiles: Fast analytics on semi-structured data. In Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), Virtual Event, China, 20–25 June 2021; pp. 445–458. [CrossRef]
2. Belefqih, S.; Barchane, M.; Zellou, A.; Benlahmar, E.H. Schema validation and evaluation framework for extracted schemas in JSON databases. *Sci. Rep.* **2026**, *16*, 10873. [CrossRef]
3. Rush, E.N.; Danciu, I.; Ostrouchov, G.; Cho, K.; Mayer, B.W.; Ho, Y.-L.; Honerlaw, J.; Costa, L.; Linares, F.; Begoli, E.; et al. JSONize: A scalable machine learning pipeline to model medical notes as semi-structured documents. *AMIA Jt. Summits Transl. Sci. Proc.* **2020**, *2020*, 533–541.
4. Carvalho, I.; Sá, F.; Bernardino, J. Performance evaluation of NoSQL document databases: Couchbase, CouchDB, and MongoDB. *Algorithms* **2023**, *16*, 78. [CrossRef]
5. Priya, D.U.; Thilagam, P.S. JSON document clustering based on schema embeddings. *J. Inf. Sci.* **2024**, *50*, 1112–1130. [CrossRef]
6. Petit, A.; Damigos, M.; D'Orazio, L.; Kalogeros, E. Computation of fuzzy joins over large collections of JSON data using semantic similarity. In Proceedings of the 28th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP 2026), Tampere, Finland, 24 March 2026; CEUR Workshop Proceedings; Volume 4186, pp. 107–115. Available online: https://ceur-ws.org/Vol-4186/short3.pdf (accessed on 14 October 2026).
7. Wang, Y.; DeWitt, D.J.; Cai, J.-Y. X-Diff: An effective change detection algorithm for XML documents. In Proceedings of the 19th International Conference on Data Engineering (ICDE '03'), Bangalore, India, 5–8 March 2003; pp. 519–530. [CrossRef]
8. Hütter, T.; Augsten, N.; Kirsch, C.M.; Carey, M.J.; Li, C. JEDI: These aren't the JSON documents you're looking for... In Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), Philadelphia, PA, USA, 12–17 June 2022; pp. 1584–1597. [CrossRef]
9. Curavale, D.; Nicolae, G.; Caranica, A.; Cucu, H.; Burileanu, C.; Davidoiu, V.; Buzo, A.; Pelz, G. Efficient similarity-based datasheet retrieval and analysis using retrieval-augmented generation for electronic component selection. *Electronics* **2026**, *15*, 2301. [CrossRef]
10. Smajić, A.; Karlović, R.; Bobanović Dasko, M.; Lorencin, I. Large language models for structured and semi-structured data, recommender systems and knowledge base engineering: A survey of recent techniques and architectures. *Electronics* **2025**, *14*, 3153. [CrossRef]
11. Zhao, Z.-L.; Zhang, R.-F. Artificial Intelligence-Enhanced Mathematical Derivation method: Exact solutions of the Benjamin–Bona–Mahony equation. *Eng. Appl. Artif. Intell.* **2026**, *173*, 114464. [CrossRef]
12. Wang, J.; Liu, Y.; Yan, L.; Han, K.; Feng, L.; Zhang, R. Fractional sub-equation neural networks (fSENNs) method for exact solutions of space–time fractional partial differential equations. *Chaos* **2025**, *35*, 043110. [CrossRef]
13. Tai, K.C. The tree-to-tree correction problem. *J. ACM* **1979**, *26*, 422–433. [CrossRef]
14. Bille, P. A survey on tree edit distance and related problems. *Theor. Comput. Sci.* **2005**, *337*, 217–239. [CrossRef]
15. Cobéna, G.; Abiteboul, S.; Marian, A. Detecting changes in XML documents. In Proceedings of the 18th International Conference on Data Engineering (ICDE '02), San Jose, CA, USA, 26 February–1 March 2002; pp. 41–52. [CrossRef]
16. Pawlik, M.; Augsten, N. RTED: A robust algorithm for the tree edit distance. *Proc. VLDB Endow.* **2011**, *5*, 334–345. [CrossRef]
17. Pan, Z.; Pan, G.; Monti, A. Semantic-similarity-based schema matching for management of building energy data. *Energies* **2022**, *15*, 8894. [CrossRef]
18. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186. [CrossRef]

19. Hua, Y.; Yang, Y.; Du, J. Deep multi-modal metric learning with multi-scale correlation for image-text retrieval. *Electronics* **2020**, *9*, 466. [CrossRef]
20. Gull, S.; Kim, J. Metric-based meta-learning approach for few-shot classification of brain tumors using magnetic resonance images. *Electronics* **2025**, *14*, 1863. [CrossRef]
21. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **2009**, *20*, 61–80. [CrossRef]
22. Kuhn, H.W. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97. [CrossRef]
23. Meta. Llama 3.2 Model Card. 2024. Available online: https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md (accessed on 14 October 2026).
24. HL7. FHIR Release 5 (R5), Version 5.0.0. Available online: https://hl7.org/fhir/R5/ (accessed on 14 October 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.