UniSAGE: Unifying Static and Dynamic Attributes with Hyper-Structure

Taoran Fang[1] , **Yan Deng**[1] , **Chunping Wang**[2] , **Yang Wang**[2] , **Lei Chen**[2] and **Yang Yang**[1*]
[1]Zhejiang University
[2]FinVolution Group
{fangtr, dyan, yangya}@zju.edu.cn, {wangchunping02, wangyang09, chenlei04}@xinye.com

Abstract

With the rapid growth of digital data, real-world applications increasingly involve hierarchical information that combines static attributes with dynamic records. Modeling such heterogeneous data in a unified and generalizable manner remains challenging. Existing approaches often rely on extensive manual design, are tightly coupled to specific data schemas, and typically process static and dynamic attributes in isolation, thereby overlooking their implicit interactions. We propose UniSAGE, a unified framework for modeling data with both static and dynamic attributes. UniSAGE constructs a global attribute graph that represents hierarchical and temporal relationships in a unified structure. To ensure representational consistency, it introduces two orthogonal parameter subspaces that jointly support static aggregation and dynamic reasoning within a shared semantic space. Building on these unified representations, UniSAGE further enables task-specific interaction between static and dynamic attributes via a lightweight hyper-structure mechanism. UniSAGE is fully automated, robust to evolving data schemas, and capable of capturing complex cross-attribute dependencies. Extensive experiments on multiple public benchmarks and a real-world financial behavior dataset demonstrate that UniSAGE consistently outperforms existing methods, achieving performance improvements of over 10% on several tasks. Our code is available at https://github.com/zjunet/UniSAGE.

1 Introduction

In the era of big data, modern information systems generate and collect data that are increasingly diverse, heterogeneous, and large-scale. From financial transactions and e-commerce activities to healthcare records and social behaviors, such data are used to model a wide range of real-world entities. As data sources continue to grow in both complexity and volume, efficiently extracting meaningful representations to support downstream tasks remains a fundamental challenge [Wang *et al.*, 2018; Xiao *et al.*, 2022].

This paper focuses on feature extraction from a class of complex data that we refer to as hierarchical data with both static and dynamic components. Such data typically comprise static attributes (*e.g.*, age, gender, product category) together with dynamic records (*e.g.*, user actions, transaction sequences, event logs). In practice, these heterogeneous attributes are often organized in semi-structured formats such as JSON, where entities exhibit hierarchical schemas with nested attributes and timestamped logs. This data paradigm is ubiquitous across real-world applications, including recommendation systems [Middleton *et al.*, 2004; Hossain *et al.*, 2024], healthcare records [Davis Giardina *et al.*, 2014; Agniel *et al.*, 2018], and financial transaction monitoring [Jayathilake, 2012; Sheble *et al.*, 2009]. Effectively modeling such data is critical for downstream tasks such as classification [Oza and Tumer, 2008], risk assessment [Gritzalis *et al.*, 2018], and recommendation [Javed *et al.*, 2021].

However, effectively modeling hierarchical data with static and dynamic attributes remains challenging. First, static and dynamic attributes are often modeled separately, leading to incompatible representation spaces and fragmented semantic reasoning. As a result, capturing a holistic entity representation typically requires extensive manual model engineering, which is costly and difficult to scale. Second, many existing approaches are tightly coupled to fixed data schemas, limiting their adaptability. In real-world industrial systems, data schemas frequently evolve as attributes are added or modified, yet most models require retraining from scratch to accommodate such changes. Third, capturing interactions between static and dynamic attributes is inherently difficult. In many applications, these two attribute types are strongly interdependent. For example, in financial lending, the combination of recent loan activities (dynamic) and age-related information (static) often jointly determines default risk. Nevertheless, most existing models process static and dynamic attributes in isolation, missing crucial cross-type correlations that are essential for accurate decision-making.

To address these challenges, we propose **Uni**fying **S**tatic and dynamic **A**ttributes with **G**raph **E**nhancement (UniSAGE), a unified framework that models static and dynamic attributes within a shared representation space while explicitly capturing their interactions. UniSAGE first constructs a

*Corresponding author.

global attribute graph that reflects the hierarchical structure of the original data. Static attributes and timestamped dynamic records are represented uniformly as nodes, with static links encoding hierarchical relations and dynamic links modeling temporal dependencies. This unified graph provides a structured modeling space in which information can be propagated coherently across attribute types. The representation of an entity is obtained by aggregating information from all nodes into a root node. To ensure representational consistency, UniSAGE introduces a joint parameterization scheme with two orthogonal subspaces for static aggregation and dynamic reasoning. Orthogonality prevents mutual interference between the two modeling paradigms, allowing static and dynamic attributes to be processed independently while remaining embedded in a unified semantic space. As a result, UniSAGE no longer distinguishes static and dynamic attributes at the representation level, but only through their respective propagation paths. This design enables automatic handling of heterogeneous attributes and provides strong generalizability to evolving data schemas. Building on these unified representations, UniSAGE further constructs task-specific hyper-structures to capture higher-order interactions between selected static and dynamic attributes. These hyper-structures introduce specialized reasoning paths tailored to downstream objectives. To avoid the high computational cost of explicit hyper-structure construction, we propose **S**elective **S**emantic **Agg**regation (SSAgg), a lightweight mechanism that selectively aggregates task-relevant information. We theoretically show that SSAgg can simulate arbitrary hyper-structures, and empirically demonstrate its effectiveness across diverse tasks. Overall, our contributions are summarized as follows:

- We propose UniSAGE, a unified framework that models static and dynamic attributes in a shared representation space, enabling automated and generalizable learning over hierarchical data.
- We introduce SSAgg, a theoretically grounded and computationally efficient mechanism for capturing task-specific interactions across static and dynamic attributes.
- Extensive experiments on public benchmarks and a real-world financial dataset demonstrate that UniSAGE consistently outperforms existing methods, achieving improvements of over 10% on several tasks.

2 Related Work

Graph-Based Encoding. Transforming complex or semi-structured data into graph representations is a common paradigm in data mining and representation learning [Jeon *et al.*, 2013; Patil *et al.*, 2018]. Such graph-based encodings have been widely applied in recommendation systems [Wang *et al.*, 2021], healthcare [Hassanain *et al.*, 2022; Arch-Int *et al.*, 2017], and finance [Zehra *et al.*, 2021; Zhang *et al.*, 2023; Onnela *et al.*, 2003], where rich dependencies among heterogeneous components must be modeled explicitly. Graph neural networks (GNNs) are a natural choice for learning over such graph-structured data [Wu *et al.*, 2020; Zhou *et al.*, 2020]. Static GNN models [Kipf, 2016; Hamilton *et al.*, 2017; Veličković *et al.*, 2017] focus on learning representations from fixed graph structures and are effective in capturing hierarchical or relational patterns. However, they are not designed to handle temporal evolution. To address this limitation, dynamic GNNs [Skarding *et al.*, 2021; Feng *et al.*, 2024; Yang *et al.*, 2024] extend static models by incorporating temporal information and evolving graph structures. Despite their success, most dynamic GNNs assume that temporal information is aligned across nodes or edges, or that dynamic interactions are defined over a shared timeline. This assumption is often violated in real-world scenarios involving heterogeneous attributes and asynchronous records, where different attributes may have varying numbers of events and non-aligned timestamps. Moreover, existing graph-based methods typically treat static and dynamic information using different modeling pipelines, which complicates representation alignment and parameter sharing. In contrast, UniSAGE adopts an attribute-level graph formulation in which both static attributes and dynamic records are treated as nodes. By jointly encoding hierarchical relations among attributes and temporal dependencies among records, this formulation avoids the need for strict temporal alignment and enables unified processing of heterogeneous information. In our experiments, we include representative static and dynamic GNN models as baselines to evaluate the effectiveness of this unified design.

Relational Deep Learning. Relational deep learning (RDL) [Fey *et al.*, 2023; Dwivedi *et al.*, 2025; Robinson *et al.*, 2024; Chen *et al.*, 2025] studies learning from relational data stored in tabular databases, where entities are described by both static attributes and dynamic records. RDL methods typically construct graphs by treating each table row (i.e., record) as a node and connecting nodes according to foreign-key relationships. This formulation has shown strong performance on benchmark datasets such as RelBench [Fey *et al.*, 2023]. While RDL and UniSAGE address similar data modalities, their modeling strategies differ fundamentally. RDL operates at the record level and relies on schema-defined relations between rows, which limits its ability to explicitly capture hierarchical dependencies among attributes. In contrast, UniSAGE adopts an attribute-centric modeling paradigm, where both static and dynamic attributes are explicitly represented and organized according to hierarchical and temporal relations. This design enables finer-grained reasoning over heterogeneous attributes and naturally supports recursive, multi-level representations. We evaluate UniSAGE on the RelBench benchmark and directly compare it with representative RDL methods [Fey *et al.*, 2023]. The results demonstrate that UniSAGE is more effective in modeling complex hierarchical and dynamic structures, especially in settings where attributes exhibit heterogeneous schemas and asynchronous records.

3 Preliminary

Categories of Data Attributes. In real-world applications, complex data used for modeling is commonly represented in tabular or semi-structured formats such as JSON. Attributes in such data can be broadly categorized as either *static* or *dynamic*, depending on whether their values evolve over time.

Static Attribute. A static attribute refers to a property of an entity whose value remains constant or changes negligibly over a sufficiently long period. For instance, in a credit loan system, user characteristics such as gender or place of birth are typically treated as static attributes. The value of a static attribute s, denoted as x_s, depends only on the associated entity e and is defined as

$$x_s(e) = f_s(e), \tag{1}$$

where $f_s(\cdot)$ is the mapping function corresponding to the static attribute s.

Dynamic Attribute. Dynamic attributes describe properties whose values change over time or accumulate records as time progresses. For example, a user's residential address may change over time, and a user's borrowing history grows as new loan records are added. The value of a dynamic attribute d at time t, denoted as x_d^t, depends on both the entity e and the timestamp t:

$$x_d^t(e) = f_d(e, t). \tag{2}$$

We represent a dynamic attribute as a time-indexed collection of records,

$$x_d(e) = \{(x_d^t(e), t) \mid t \in [1, T]\}, \tag{3}$$

where $f_d(\cdot)$ is the mapping function associated with attribute d, and T denotes the number of observed records. In practice, dynamic attributes may vary across entities in both sequence length and timestamp distribution. For example, different users typically have different numbers of loan records, and their timestamps are rarely aligned.

Overall Objective. Given an entity e with its associated static and dynamic attributes, our goal is to learn a representation h_e through a function ϕ:

$$h_e = \phi\big(\{x_s(e) \mid s \in \mathcal{S}\}, \{x_d(e) \mid d \in \mathcal{D}\}\big), \tag{4}$$

where $\mathcal{S}$ and $\mathcal{D}$ denote the sets of static and dynamic attributes, respectively. The learned representation h_e serves as the input to downstream tasks such as classification, regression, or user modeling.

4 Methodology

4.1 Hierarchical Graph Construction

We represent each entity instance as a directed hierarchical graph G that unifies both static attributes and dynamic records. The graph is organized as a rooted tree, where edges point from sub-attributes (or earlier records) to their parents (or later records), and the root aggregates all information into an entity embedding.

Node Generation. Each attribute is instantiated as a node following the hierarchical schema. Given an attribute a, we recursively create a node v_a and expand its sub-attributes $\mathrm{Sub}(a)$ until reaching leaf attributes:

$$g(a) = \begin{cases} v_a, & \text{if } \mathrm{Sub}(a) = \emptyset, \\ v_a \to \{g(a') \mid a' \in \mathrm{Sub}(a)\}, & \text{otherwise.} \end{cases} \tag{5}$$

For a dynamic attribute d with T timestamped records, we treat each record as an individual node $\{v_{d_t}\}_{t=1}^T$ (and recursively expand its sub-attributes if present):

$$g(d) = \begin{cases} \{v_{d_t}\}_{t=1}^T, & \text{if } \mathrm{Sub}(d_t) = \emptyset \ \forall t, \\ \{v_{d_t}\}_{t=1}^T \to \{g(d_t)\}_{t=1}^T, & \text{otherwise.} \end{cases} \tag{6}$$

Static Links. Static links encode the schema hierarchy: for each parent attribute a_j and its (static) sub-attribute $a_i \in \mathrm{Sub}(a_j)$, we add a directed edge $v_{a_i} \to v_{a_j}$. We define the static adjacency matrix $\mathbf{A}_s \in \mathbb{R}^{N \times N}$ as

$$\mathbf{A}_s(i, j) = \begin{cases} 1, & \text{if } a_i \in \mathrm{Sub}(a_j), \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

When a sub-attribute is dynamic (i.e., it corresponds to a record sequence $\{d_t\}_{t=1}^T$), connecting the parent to all records would introduce redundant paths and inflate computation. Instead, we connect the parent only to the most recent record node d_T, which preserves the latest state while keeping the hierarchy concise:

$$\mathbf{A}_s(d_T, a_j) = 1, \tag{8}$$

$$\mathbf{A}_s(d_t, a_j) = 0, \quad t < T, \quad \text{for } \{d_t\}_{t=1}^T \in \mathrm{Sub}(a_j). \tag{9}$$

Dynamic Links. Dynamic links encode temporal order within each dynamic attribute. For consecutive records of the same attribute, we add directed edges from earlier to later records:

$$\mathbf{A}_d(i, j) = \begin{cases} 1, & \text{if } a_i = d_t \text{ and } a_j = d_{t+1}, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

Finally, we add a root node r and connect each top-level attribute node to r with a directed edge, forming a rooted tree. The entity representation is obtained from the final embedding of the root node r.

4.2 Orthogonal Transformation Design

UniSAGE performs bottom-up propagation on G to compute the root representation. A key challenge is that static aggregation (hierarchical composition) and dynamic reasoning (temporal modeling) typically rely on different operators (*e.g.*, GNN/MLP vs. RNN/GRU), which can distort semantics and hinder parameter sharing even when inputs are encoded in the same embedding space. To address this, we introduce two *orthogonal* parameter subspaces: $\mathbf{W}_s$ for static aggregation and $\mathbf{W}_d$ for dynamic reasoning. Orthogonality encourages the two pathways to be non-interfering while keeping representations in a shared semantic space. Formally, we first map node features into a high-dimensional space $\mathbf{H} \in \mathbb{R}^{N \times k}$. We then constrain parameters for static and dynamic transformations to lie in the spans of $\mathbf{W}_s$ and $\mathbf{W}_d$, respectively, and enforce

$$\theta_s \in \mathrm{span}(\mathbf{W}_s),\ \theta_d \in \mathrm{span}(\mathbf{W}_d) \quad \text{s.t. } \mathbf{W}_s^\top \mathbf{W}_d = \mathbf{0}. \tag{11}$$

Here $\mathrm{span}(\mathbf{W})$ denotes the subspace spanned by columns of $\mathbf{W}$. This constraint reduces cross-talk between static and dynamic transformations, stabilizing training and enabling consistent semantics under different propagation paths.

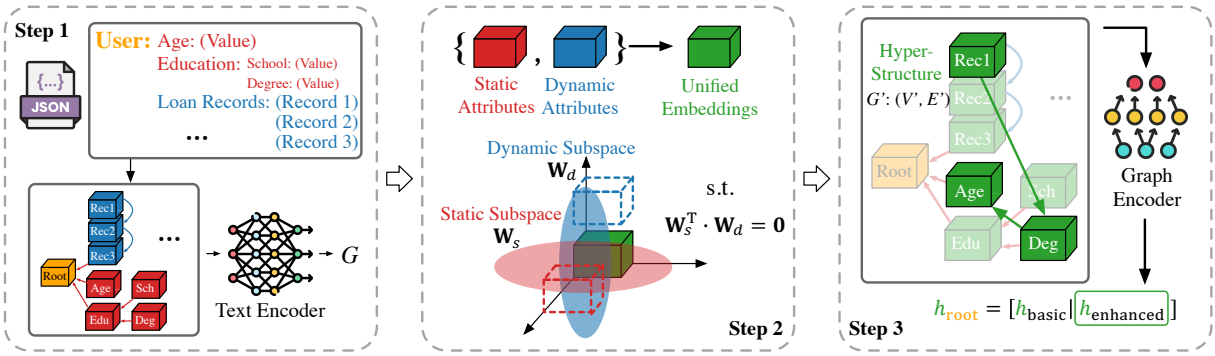

Figure 1: Overview of UniSAGE. **Step 1:** Construct a global attribute graph from JSON, with nodes as attributes/records and edges encoding hierarchical (static) and temporal (dynamic) relations. **Step 2:** Learn unified representations via orthogonal subspaces for non-interfering static aggregation and dynamic reasoning. **Step 3:** Capture task-specific cross-type interactions through a lightweight hyper-structure mechanism (SSAgg) to enhance downstream prediction.

4.3 Hyper-Structure Generation

The global graph G captures hierarchical and temporal relations but does not explicitly model *task-specific* interactions between static and dynamic attributes. However, many decisions depend on cross-type combinations (e.g., recent behaviors conditioned on demographics). To capture such interactions, we define a task-specific hyper-structure as a subgraph $G' = (\mathcal{V}', \mathcal{E}')$ induced from $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}' \subseteq \mathcal{V}$ contains task-relevant nodes and $\mathcal{E}'$ connects them to form specialized reasoning paths. We then compute a hyper-structure representation via a graph encoder f:

$$h_{G'} = f(\mathcal{V}', \mathcal{E}'), \text{ s.t. } \mathcal{V}' \subseteq \mathcal{V}. \tag{12}$$

In practice, we instantiate f with a GNN due to its strong relational reasoning ability [Xu *et al.*, 2021; Xu *et al.*, 2020b; Yang *et al.*, 2022; Yang *et al.*, 2023; Dudzik and Velickovic, 2022]. Finally, $h_{G'}$ is combined with the root representation h_r for downstream prediction.

4.4 Practical Implementation

We now describe the end-to-end implementation (Figure 1), corresponding to Sections 4.1–4.3.

Step 1: Global graph and node features. We construct G using $\mathbf{A}_s$ and $\mathbf{A}_d$ as in Section 4.1. Each node is associated with a text input that describes its attribute/value. Specifically, for leaf nodes we use "attribute name: value"; for intermediate attribute nodes we use "attribute name: [attribute flag]"; and for the root node we use a special marker "[root flag]". We encode each node text with a pretrained text encoder:

$$x_{\mathrm{raw}}(v) = \begin{cases} \text{"attribute name: value"}, & \text{if } v \text{ is a leaf node,} \\ \text{"[root flag]"}, & \text{if } v \text{ is the root node,} \\ \text{"attribute name: [attribute flag]"}, & \text{otherwise,} \end{cases}$$

$$x(v) = \mathrm{TextEncoder}\big(x_{\mathrm{raw}}(v)\big), \tag{13}$$

yielding the node feature matrix $\mathbf{X} \in \mathbb{R}^{N \times k}$.

Step 2: Unified propagation with orthogonal subspaces. We first project $\mathbf{X}$ into $\mathbf{H}$:

$$\mathbf{H} = \mathbf{X}\mathbf{W}, \tag{14}$$

where $\mathbf{W} \in \mathbb{R}^{k \times k'}$ is learnable. We then define two subspace bases $\mathbf{W}_s \in \mathbb{R}^{k' \times k_s}$ and $\mathbf{W}_d \in \mathbb{R}^{k' \times k_d}$ and enforce orthogonality:

$$\mathbf{W}_s^\top \mathbf{W}_d = \mathbf{0}. \tag{15}$$

We implement this constraint by adding the orthogonality loss

$$l_o = \|\mathbf{W}_s^\top \mathbf{W}_d\|_F^2, \tag{16}$$

while noting that numerical orthogonalization (e.g., Gram–Schmidt) is also applicable [Björck, 1994; Leon *et al.*, 2013; Björck, 1967].

For static links $\mathbf{A}_s$, we perform hierarchical aggregation where messages are mainly transformed in the static subspace while retaining complementary information from the dynamic subspace:

$$h_j^* = \Big(\alpha\, h_j \mathbf{W}_s + \sum_{\mathbf{A}_s(i,j)=1} \beta_i\, h_i^* \mathbf{W}_s\Big)\mathbf{W}_s^+ + \frac{1}{n+1}\Big(h_j \mathbf{W}_d + \sum_{\mathbf{A}_s(i,j)=1} h_i^* \mathbf{W}_d\Big)\mathbf{W}_d^+, \tag{17}$$

where h is the current representation, h^* is the updated representation (with $h^* = h$ for leaf nodes), n is the number of children of v_j, and $\alpha, \{\beta_i\}$ are learnable aggregation weights (analogous to attention weights in GAT [Veličković *et al.*, 2017]). $\mathbf{W}^+$ denotes the pseudo-inverse of $\mathbf{W}$.

For dynamic links $\mathbf{A}_d$, we use a temporal encoder (GRU by default) to propagate along time:

$$h_j^* = \underset{\mathbf{A}_d(i,j)=1}{T_{\mathrm{encoder}}}\big(\mathrm{state}(i),\, h_j \mathbf{W}_d\big)\mathbf{W}_d^+ + \frac{1}{2}(h_j + h_i^*)\mathbf{W}_s\mathbf{W}_s^+, \tag{18}$$

where $\mathrm{state}(i)$ is the hidden state carried from the previous record. Eqs. 17–18 implement non-interfering static/dynamic processing: static aggregation primarily operates in $\mathbf{W}_s$ and dynamic reasoning in $\mathbf{W}_d$, while both preserve complementary information to maintain a shared semantic space. We apply these updates bottom-up from leaves to the root and obtain h_r^* as the basic entity representation:

$$h_{\mathrm{basic}} = h_r^*. \tag{19}$$

Step 3: Lightweight hyper-structure via SSAgg. Explicitly selecting $\mathcal{V}'$ and constructing $\mathcal{E}'$ can be expensive. We therefore propose **S**elective **S**emantic **Agg**regation (SSAgg), which implicitly simulates task-specific hyper-structures by re-propagating messages over a unified adjacency matrix:

$$\mathbf{A}' = \mathbf{A}_s + \mathbf{A}_d \in \mathbb{R}^{N \times N}. \tag{20}$$

SSAgg computes attention scores and selectively aggregates incoming messages:

$$m_{i,j} = \psi(h_i^*, h_j^*), \quad \text{for } \mathbf{A}'(i,j) = 1, \tag{21}$$

$$\beta_{i,j} = \frac{\exp(m_{i,j})}{\exp(\lambda) + \sum_{\mathbf{A}'(k,j)=1} \exp(m_{k,j})}, \tag{22}$$

$$h'_j = \sum_{\mathbf{A}'(i,j)=1} \beta_{i,j}\, h_i^* \mathbf{W}', \tag{23}$$

where ψ is a learnable scoring function, $\mathbf{W}'$ is learnable, and λ controls selective suppression. Compared with standard softmax attention, $\exp(\lambda)$ in the denominator provides a tunable "gate" that can down-weight all incoming messages, enabling SSAgg to mimic node/edge selection (formalized in Section 4.5). We denote the resulting root representation as

$$h_{\text{enhanced}} = h'_r. \tag{24}$$

Training objective. We concatenate the basic and enhanced representations for prediction:

$$h_{\text{final}} = [h_{\text{basic}} \| h_{\text{enhanced}}], \tag{25}$$

and optimize the downstream loss with orthogonality regularization:

$$l = l_{\text{downstream}} + \gamma\, l_o, \tag{26}$$

where γ balances the orthogonality constraint.

4.5 Theoretical Guarantee

In this section, we provide theoretical justification for the effectiveness of SSAgg. Recall that explicit hyper-structure construction typically involves two expensive steps: selecting task-relevant nodes and generating new edges among them. SSAgg circumvents both steps by selectively suppressing or propagating messages over the original graph. We formalize this intuition through the following propositions.

Proposition 1. *Given a central node v_c, a set of neighbor nodes $\{v_i\}_{i=1}^n$ with directed edges pointing to v_c, and any fixed value of λ, there exists a scoring function ψ of the form defined in Formula 21 such that*

$$\forall v_k \in \{v_i\}_{i=1}^n,\ \forall \delta > 0,\ \beta_{k,c} < \delta, \tag{27}$$

where $\beta_{k,c}$ denotes the aggregation coefficient of node v_k computed according to Formula 22.

The proof is provided in Appendix A.1. Proposition 1 shows that, by introducing the hyperparameter λ, SSAgg can arbitrarily suppress the contribution of any selected neighbor node. In other words, SSAgg can effectively block message propagation from specific nodes. This selective suppression is functionally equivalent to node selection in explicit hyper-structure construction.

We next extend this result to show that SSAgg is expressive enough to match the root-level representations produced by a broad class of hyper-structure GNNs. Specifically, we demonstrate that SSAgg can reproduce the representation obtained by applying any graph neural network to any given hyper-structure graph.

Proposition 2. *Given the original graph $G = (\mathcal{V}, \mathcal{E})$ and an arbitrary hyper-structure graph $G' = (\mathcal{V}', \mathcal{E}')$ with $\mathcal{V}' \subseteq \mathcal{V}$, for any graph neural network f used to encode G', there exists a set of SSAgg parameters θ such that*

$$\underset{\theta}{\text{SSAgg}}(G) = h'_r = f(\mathcal{V}', \mathcal{E}'), \tag{28}$$

where $\text{SSAgg}(\cdot)$ denotes the process of generating the enhanced root representation h'_r in Step 3.

The detailed proof is provided in Appendix A.2. Together, Propositions 1 and 2 establish that SSAgg can implicitly realize both node selection and structure construction. Consequently, SSAgg is theoretically guaranteed to match the representational power of explicitly constructed hyper-structures, while operating directly on the original graph with substantially lower computational overhead.

5 Experiment

In this section, we systematically evaluate the effectiveness of UniSAGE across multiple tasks. Experiments are conducted on both public benchmarks and a real-world industrial dataset. We also perform comprehensive ablation studies to analyze the contribution of each component in UniSAGE.

5.1 Overall Setup

All raw data are stored in JSON format. We compare UniSAGE with four categories of baseline methods:

- Traditional Feature Extraction Models: we include traditional models *MLP* [Hornik, 1991] and *LightGBM* [Ke *et al.*, 2017] as baselines. These models directly filter and extract features from the input to meet the requirements of downstream tasks.
- Static Graph Neural Networks: we also compare UniSAGE with widely-used static GNN models including *GCN* [Kipf, 2016], *GraphSAGE* [Hamilton *et al.*, 2017] and *GAT* [Veličković *et al.*, 2017]. These methods operate on a fixed graph structure.
- Dynamic Graph Neural Networks: to evaluate performance in evolving scenarios, we include dynamic GNNs such as *TGN* [Rossi *et al.*, 2020], *TGAT* [Xu *et al.*, 2020a], *DyGFormer* [Yu *et al.*, 2023] and *FreeDyG* [Tian *et al.*, 2024]. These models capture temporal dependencies and structural changes over time.
- Relational Deep Learning: we also consider *RDL* [Fey *et al.*, 2023] and *RelGNN* [Chen *et al.*, 2025] as baselines, which are designed to model complex structured relationships in data.

Detailed descriptions of all baselines are provided in Appendix B.2. For traditional feature extraction models, we use a language model to encode JSON text into representations,

followed by task-specific predictors. For static GNN baselines, we construct a graph following Section 4.4 (Step 1), obtain node features via a language model, and apply GNNs on the combined adjacency matrix $\mathbf{A} = \mathbf{A}_s + \mathbf{A}_d$. Dynamic GNNs follow a similar construction, treating dynamic attributes as time-evolving nodes. For RDL-based models, we strictly follow their original graph construction and learning procedures. For UniSAGE, we adopt GRU [Chung *et al.*, 2014] as the temporal encoder in Formula 18. Hyperparameters and implementation details are reported in Appendix B.1.

5.2 Results on RelBench

Dataset Introduction. RelBench [Robinson *et al.*, 2024] is a public benchmark for learning on relational databases, covering domains such as e-commerce, social networks, and healthcare, with dataset sizes ranging from 74K to 41M entities. We convert all tabular data into JSON format; details are provided in Appendix B.3. Due to the large scale of certain datasets, some baselines cannot be evaluated on full data. Following prior practice, we uniformly resample 2,000/500/500 instances for training/validation/testing across all tasks. Full-dataset results of UniSAGE are reported in Appendix B.6.

Entity Classification Results. Table 1 reports the AUC-ROC results on all entity classification tasks. We summarize the key observations as follows.

1. UniSAGE consistently outperforms all baselines. UniSAGE achieves the best results on all evaluated tasks. On average, it improves over traditional feature-based models by 15.93% and over GNN-based methods by 9.22%. In several tasks, UniSAGE exceeds the second-best method by nearly 10%, demonstrating strong and stable performance across diverse datasets.

2. UniSAGE robustly integrates structural and temporal information. While static GNNs outperform traditional baselines by 6.71% on average, their performance varies significantly across tasks. Dynamic GNNs introduce temporal signals but yield only marginal gains over static counterparts (0.52% on average), and even underperform in some cases. In contrast, UniSAGE outperforms dynamic GNNs by 8.95% on average, indicating its superior ability to jointly model heterogeneous structures and asynchronous temporal behaviors.

3. UniSAGE generalizes beyond schema-specific relational models. Compared with RDL and RelGNN, which are tailored to RelBench schemas, UniSAGE achieves an average improvement of nearly 5%. For certain tasks, such as rel-f1 driver-dnf, the gain exceeds 10%, highlighting UniSAGE's strong generalization capability on complex relational data.

Entity Regression Results. Results on entity regression tasks are reported in Appendix B.5. UniSAGE consistently achieves the best performance across all datasets, outperforming traditional models, static GNNs, dynamic GNNs, and RDL methods by 22.34%, 14.05%, 10.87%, and 8.28% on average, respectively. These results further confirm the robustness of UniSAGE across different learning objectives.

5.3 Results on Real-World Data

Dataset Introduction. We use a large-scale real-world financial behavior dataset, referred to as **UserBehavior**, which contains 191,927 user records with both demographic attributes (*e.g.*, gender, education, employment status) and temporal behavioral logs (*e.g.*, loan applications, credit account transactions). The dataset is stored in JSON format, where each user profile consists of a set of static features along with a sequence of dynamic events. The downstream task is formulated as a binary anomaly-detection problem, aiming to predict whether a user poses a potential credit risk. We use 115,156 records (60%) for training, 30,097 records (15%) for validation, and 46,674 records (25%) for testing. All baselines follow the same preprocessing protocol as in RelBench, and we use the same text encoder [Reimers and Gurevych, 2019] for feature construction. To ensure data privacy and regulatory compliance, the dataset is rigorously anonymized and all sensitive information is removed. Although the dataset is not publicly available at present, a public release is under consideration. Detailed dataset statistics are provided in Appendix B.7.

Experimental Results. Table 2 reports the performance of all baselines on UserBehavior. The results indicate that this task is challenging: simple text encoding followed by an MLP fails to produce a competitive model. Incorporating graph structure yields measurable gains, but the improvements remain modest, suggesting that existing static GNNs struggle to fully exploit the underlying hierarchical organization. Dynamic GNNs that model temporal semantics provide additional benefits, yet the overall performance is still limited, indicating that effectively leveraging asynchronous behavioral logs remains difficult. In contrast, UniSAGE achieves a substantial performance boost. Compared to the best competing baseline (DyGFormer), UniSAGE delivers a relative improvement of nearly 10%. These results underscore the advantage of UniSAGE's unified modeling of static and dynamic attributes, demonstrating its suitability for complex real-world industrial data and its ability to extract task-relevant signals in challenging scenarios.

5.4 Ablation Study

We conduct ablation studies to quantify the contribution of each component in UniSAGE. Specifically, we evaluate on the rel-f1 driver-dnf task and the UserBehavior dataset, and compare the following variants:

1. UniSAGE without unified representation and without SSAgg (UniSAGE w/o UR and w/o SSAgg): This variant uses only Step 1 in Section 4.4 to construct the graph and perform bottom-up propagation from leaf nodes to the root. It removes unified representation learning in Step 2 and does not apply SSAgg for representation enhancement.

2. UniSAGE without orthogonality loss and without SSAgg (UniSAGE w/o OL and w/o SSAgg): Based on Variant 1, this model includes the parameter subspaces $\mathbf{W}_s$ and $\mathbf{W}_d$ in Step 2 but removes the orthogonality loss in Formula 16, i.e., no constraint is imposed to enforce orthogonality between the two spaces. SSAgg is also excluded.

Dataset	rel-amazon		rel-avito		rel-event		rel-f1		rel-hm	rel-stack		rel-trial
Task	user-churn	item-churn	user-visits	user-clicks	user-repeat	user-ignore	driver-dnf	driver-top3	user-churn	user-engage	user-badge	study-outcome
MLP	50.24	50.79	54.92	50.64	66.26	72.83	66.30	64.53	50.05	57.61	66.72	52.78
LightGBM	50.36	50.84	57.74	51.34	67.05	77.58	70.35	70.09	50.66	66.69	67.01	53.87
GCN	51.17	70.71	60.59	57.53	67.96	78.28	74.51	74.76	51.83	73.48	73.34	55.45
GraphSAGE	51.96	70.81	61.17	56.39	67.09	78.26	71.82	74.99	53.57	70.32	74.92	54.73
GAT	52.14	73.42	61.54	62.41	67.42	80.77	75.39	76.11	57.56	79.31	79.61	56.12
TGN	51.68	65.10	58.29	52.65	67.45	80.48	70.70	70.25	69.71	71.80	80.45	55.46
TGAT	56.19	56.70	62.51	52.95	67.30	79.32	73.24	73.80	65.65	76.32	88.34	54.16
DyGFormer	52.11	57.12	58.69	59.77	67.66	77.70	75.71	72.37	65.15	80.08	86.81	56.00
FreeDyG	51.54	54.52	65.34	65.75	68.87	80.52	71.51	70.61	65.22	80.13	83.20	54.48
RDL	58.06	81.43	68.27	71.09	69.91	80.37	70.35	73.03	68.63	86.52	84.16	53.34
RelGNN	55.70	80.80	65.80	68.57	68.01	81.37	70.90	73.60	66.26	82.19	84.50	51.27
UniSAGE	**59.24**	**81.69**	**70.60**	**71.88**	**70.38**	**87.05**	**82.57**	**82.20**	**71.40**	**88.92**	**90.21**	**57.17**

Table 1: Entity classification results (AUC-ROC % ↑, higher is better) on RelBench datasets. The optimal results are **bolded**, and the sub-optimal results are underlined.

Model	MLP	LightGBM	GCN	GraphSAGE	GAT
Result	50.10	52.12	52.64	52.34	53.80
Model	TGN	DyGFormer	TGAT	FreeDyG	UniSAGE
Result	53.91	54.61	54.03	54.12	**59.06**

Table 2: The experimental results (AUC-ROC % ↑, higher is better) on the real-world industrial dataset UserBehavior.

3. UniSAGE without SSAgg (UniSAGE w/o SSAgg): This variant includes both Step 1 and Step 2 but removes SSAgg, i.e., it does not perform representation enhancement via hyper-structure reasoning.

4. UniSAGE : The full model that includes all components described in our framework.

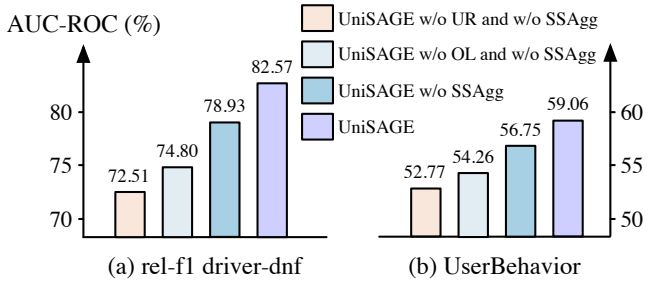

Figure 2: The ablation results for UniSAGE.

Figure 2 presents the ablation results. Using only the graph structure yields clearly suboptimal performance (UniSAGE w/o UR and w/o SSAgg), indicating that naive propagation is insufficient. Introducing separate static aggregation and dynamic inference improves performance (UniSAGE w/o OL and w/o SSAgg), highlighting the importance of modeling both attribute types. However, without the orthogonality loss, the static and dynamic representation spaces may drift during training, which limits the effectiveness of this unified modeling. Adding the orthogonality constraint stabilizes the shared semantic space and further improves performance (UniSAGE w/o SSAgg). Finally, incorporating SSAgg enables task-relevant hyper-structure reasoning and strengthens cross-type interactions, leading to the best overall results (UniSAGE). Overall, these results confirm the necessity of each component and align well with our theoretical motivation and empirical design choices.

5.5 Additional Results and Analysis

To improve readability and focus in the main paper, we defer several complementary experimental results and analyses to the appendix. Specifically, Appendix B.4 provides detailed statistics of all datasets and tasks used in RelBench. Complete results on the full (non-resampled) RelBench datasets are reported in Appendix B.6. Appendix B.8 presents a detailed empirical analysis of the runtime efficiency of UniSAGE, including comparisons across different datasets and task settings. Appendix B.9 provides an in-depth hyperparameter sensitivity study, analyzing the impact of key hyperparameters and demonstrating the robustness of UniSAGE under reasonable configurations.

6 Conclusion

In this paper, we propose UniSAGE, a unified framework for modeling complex data with both static and dynamic attributes. UniSAGE addresses key limitations of existing approaches by constructing a global attribute graph, learning unified representations through orthogonal parameter subspaces, and leveraging hyper-structures to enhance downstream tasks. Extensive experiments on public benchmarks and a real-world industrial dataset demonstrate that UniSAGE consistently outperforms strong baselines, achieving improvements of up to 10% on challenging tasks. By combining automation, strong generalizability, and effective modeling of implicit static–dynamic correlations, UniSAGE provides a practical and robust solution for real-world data-intensive applications.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 62322606, No. 62441605), and the Fundamental Research Funds for the Central Universities.

References

[Agniel *et al.*, 2018] Denis Agniel, Isaac S Kohane, and Griffin M Weber. Biases in electronic health record data due to processes within the healthcare system: retrospective observational study. *Bmj*, 361, 2018.

[Arch-Int *et al.*, 2017] Ngamnij Arch-Int, Somjit Arch-Int, Suphachoke Sonsilphong, and Paweena Wanchai. Graph-based semantic web service composition for healthcare data integration. *Journal of healthcare engineering*, 2017(1):4271273, 2017.

[Björck, 1967] Åke Björck. Solving linear least squares problems by gram-schmidt orthogonalization. *BIT Numerical Mathematics*, 7(1):1–21, 1967.

[Björck, 1994] Åke Björck. Numerics of gram-schmidt orthogonalization. *Linear Algebra and Its Applications*, 197:297–316, 1994.

[Chen *et al.*, 2025] Tianlang Chen, Charilaos Kanatsoulis, and Jure Leskovec. Relgnn: Composite message passing for relational deep learning. *arXiv preprint arXiv:2502.06784*, 2025.

[Chung *et al.*, 2014] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[Davis Giardina *et al.*, 2014] Traber Davis Giardina, Shailaja Menon, Danielle E Parrish, Dean F Sittig, and Hardeep Singh. Patient access to medical records and healthcare outcomes: a systematic review. *Journal of the American Medical Informatics Association*, 21(4):737–741, 2014.

[Dudzik and Velickovic, 2022] Andrew Dudzik and Petar Velickovic. Graph neural networks are dynamic programmers. *ArXiv*, 2022.

[Dwivedi *et al.*, 2025] Vijay Prakash Dwivedi, Charilaos Kanatsoulis, Shenyang Huang, and Jure Leskovec. Relational deep learning: Challenges, foundations and next-generation architectures. *arXiv preprint arXiv:2506.16654*, 2025.

[Feng *et al.*, 2024] ZhengZhao Feng, Rui Wang, TianXing Wang, Mingli Song, Sai Wu, and Shuibing He. A comprehensive survey of dynamic graph neural networks: Models, frameworks, benchmarks, experiments and challenges. *arXiv preprint arXiv:2405.00476*, 2024.

[Fey *et al.*, 2023] Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. Relational deep learning: Graph representation learning on relational databases. *arXiv preprint arXiv:2312.04615*, 2023.

[Gritzalis *et al.*, 2018] Dimitris Gritzalis, Giulia Iseppi, Alexios Mylonas, and Vasilis Stavrou. Exiting the risk assessment maze: A meta-survey. *ACM Computing Surveys (CSUR)*, 51(1):1–30, 2018.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[Hassanain *et al.*, 2022] Amr A Hassanain, Mohamed AA Eldosoky, and Ahmed M Soliman. Evaluating building performance in healthcare facilities using entropy and graph heuristic theories. *Scientific Reports*, 12(1):8973, 2022.

[Hornik, 1991] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[Hossain *et al.*, 2024] Ismail Hossain, Sai Puppala, Md Jahangir Alam, and Sajedul Talukder. Socialrec: User activity based post weighted dynamic personalized post recommendation system in social media. In *International Conference on Advances in Social Networks Analysis and Mining*, pages 263–280. Springer, 2024.

[Javed *et al.*, 2021] Umair Javed, Kamran Shaukat, Ibrahim A Hameed, Farhat Iqbal, Talha Mahboob Alam, and Suhuai Luo. A review of content-based and context-based recommendation systems. *International Journal of Emerging Technologies in Learning (iJET)*, 16(3):274–306, 2021.

[Jayathilake, 2012] Dileepa Jayathilake. Towards structured log analysis. In *2012 Ninth International Conference on Computer Science and Software Engineering (JCSSE)*, pages 259–264. IEEE, 2012.

[Jeon *et al.*, 2013] Seungwoo Jeon, Yohanes Khosiawan, and Bonghee Hong. Making a graph database from unstructured text. In *2013 IEEE 16th International Conference on Computational Science and Engineering*, pages 981–988. IEEE, 2013.

[Ke *et al.*, 2017] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[Kipf, 2016] TN Kipf. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Leon *et al.*, 2013] Steven J Leon, Åke Björck, and Walter Gander. Gram-schmidt orthogonalization: 100 years and more. *Numerical Linear Algebra with Applications*, 20(3):492–532, 2013.

[Middleton *et al.*, 2004] Stuart E Middleton, Nigel R Shadbolt, and David C De Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):54–88, 2004.

[Onnela *et al.*, 2003] Jukka-Pekka Onnela, Anirban Chakraborti, Kimmo Kaski, Janos Kertesz, and Antti

Kanto. Asset trees and asset graphs in financial markets. *Physica Scripta*, 2003(T106):48, 2003.

[Oza and Tumer, 2008] Nikunj C Oza and Kagan Tumer. Classifier ensembles: Select real-world applications. *Information fusion*, 9(1):4–20, 2008.

[Patil *et al.*, 2018] NS Patil, P Kiran, NP Kiran, and Naresh Patel KM. A survey on graph database management techniques for huge unstructured data. *International Journal of Electrical and Computer Engineering*, 8(2):1140, 2018.

[Reimers and Gurevych, 2019] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[Robinson *et al.*, 2024] Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan Eric Lenssen, Yiwen Yuan, Zecheng Zhang, et al. Relbench: A benchmark for deep learning on relational databases. *Advances in Neural Information Processing Systems*, 37:21330–21341, 2024.

[Rossi *et al.*, 2020] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[Sheble *et al.*, 2009] Laura Sheble, Barbara Wildemuth, and Kathy Brennan. Transaction logs. *Applications of social research methods to questions in information and library science*, pages 166–177, 2009.

[Skarding *et al.*, 2021] Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *iEEE Access*, 9:79143–79168, 2021.

[Tian *et al.*, 2024] Yuxing Tian, Yiyan Qi, and Fan Guo. Freedyg: Frequency enhanced continuous-time dynamic graph model for link prediction. In *The twelfth international conference on learning representations*, 2024.

[Veličković *et al.*, 2017] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[Wang *et al.*, 2018] Ruili Wang, Wanting Ji, Mingzhe Liu, Xun Wang, Jian Weng, Song Deng, Suying Gao, and Chang-an Yuan. Review on mining data from multiple data sources. *Pattern Recognition Letters*, 109:120–128, 2018.

[Wang *et al.*, 2021] Shoujin Wang, Liang Hu, Yan Wang, Xiangnan He, Quan Z Sheng, Mehmet A Orgun, Longbing Cao, Francesco Ricci, and Philip S Yu. Graph learning based recommender systems: A review. *arXiv preprint arXiv:2105.06339*, 2021.

[Wu *et al.*, 2020] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[Xiao *et al.*, 2022] Kejing Xiao, Zhaopeng Qian, and Biao Qin. A survey of data representation for multi-modality event detection and evolution. *Applied Sciences*, 12(4):2204, 2022.

[Xu *et al.*, 2020a] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*, 2020.

[Xu *et al.*, 2020b] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? *ICLR*, 2020.

[Xu *et al.*, 2021] Keyulu Xu, Mozhi Zhang, Jingling Li, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. *ICLR*, 2021.

[Yang *et al.*, 2022] Chenxiao Yang, Qitian Wu, Jiahua Wang, and Junchi Yan. Graph neural networks are inherently good generalizers: Insights by bridging gnns and mlps. *ArXiv*, abs/2212.09034, 2022.

[Yang *et al.*, 2023] Chenxiao Yang, Qitian Wu, David Wipf, Ruoyu Sun, and Junchi Yan. How graph neural networks learn: Lessons from training dynamics in function space. *ArXiv*, 2023.

[Yang *et al.*, 2024] Leshanshui Yang, Clement Chatelain, and Sebastien Adam. Dynamic graph representation learning with neural networks: A survey. *Ieee Access*, 12:43460–43484, 2024.

[Yu *et al.*, 2023] Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. Towards better dynamic graph learning: New architecture and unified library. *Advances in Neural Information Processing Systems*, 36:67686–67700, 2023.

[Zehra *et al.*, 2021] Samreen Zehra, Syed Farhan Mohsin Mohsin, Shaukat Wasi, Syed Imran Jami, Muhammad Shoaib Siddiqui, and Muhammad Khaliq-Ur-Rahman Raazi Syed. Financial knowledge graph based financial report query system. *IEEE Access*, 9:69766–69782, 2021.

[Zhang *et al.*, 2023] Qiuyue Zhang, Yunfeng Zhang, Xunxiang Yao, Shilong Li, Caiming Zhang, and Peide Liu. A dynamic attributes-driven graph attention network modeling on behavioral finance for stock prediction. *ACM Transactions on Knowledge Discovery from Data*, 18(1):1–29, 2023.

[Zhou *et al.*, 2020] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.